Department of Computer Science & Engineering

Bangladesh University of Engineering & Technology

CSE406 - Computer Security Sessional

Report On Snort

September 13, 2023

Supervisor:

Abdur Rashid Tushar
Lecturer

Prepared By:

Fabiha Tasneem
1805072
Level-4, Term-1

Sumaiya Sultana
1805079
Level-4, Term-1

Abstract

Snort, created by Martin Roesch in 1998, is a versatile IDS/IPS that offers real-time analysis of network traffic to detect and respond to potentially malicious activities. It operates on a rule-based system, where users can define custom rules or employ pre-existing rule sets to identify and mitigate threats. Snort's rule-based approach allows it to monitor network traffic at both the network and application layers, thereby offering comprehensive protection against a wide range of network-based attacks and vulnerabilities.

Snort represents a vital asset in the arsenal of network security tools. Its rule-based detection, adaptability, and open-source philosophy have established it as a trusted solution for identifying and mitigating network threats. As cyber threats continue to evolve, Snort's role in defending against these threats remains indispensable, ensuring the resilience of modern network infrastructures.

Contents

List of Figures

1 Introduction

1.1 What Is Snort?

Snort is a popular free and open-source IDS/ IPS system that is used to perform traffic/ protocol analysis and content matching. Snort can be used to detect and prevent various attacks based on *predefined rules.*

1.2 Snort Operations

- **Packet Sniffing:** Analyzes the actual network traffic in real-time
- **Network Intrusion Detection:** Analyzes packets and matches traffic against signatures
- **Packet Logging:** Collects and logs network traffic into a log file
- **Network Intrusion Prevention:** Takes specific actions to block identified threats

1.3 How Does Snort Work?

Snort detects malicious traffic or attacks by leveraging pattern matching. When active, Snort captures packets, reassembles them, analyzes them, and determines what needs to be done to the packet based on predefined rules.
Snort has a large number of rule sets created by the community that are very useful to begin with. Snort rules are very similar to a typical firewall rule, whereby, they are used to match network activity against specific patterns or signatures and consequently make a decision as to whether to send an alert or drop the traffic (in the case of IPS).

1.4 Snort Rules

Snort has 4 types of rule sets:

- **Community Rules:** Free rule sets created by the Snort community.
- **Registered Rules:** Free rule sets created by Talos. In order to use them, a user must register for an account.
- **Subscription Only Rules:** These rule sets require an active paid subscription in order to be accessed and used.
- **Customized Rules:** We can write our own rules based on our requirements.

A Snort Rule is very simple and easy to write. The syntax goes like this:

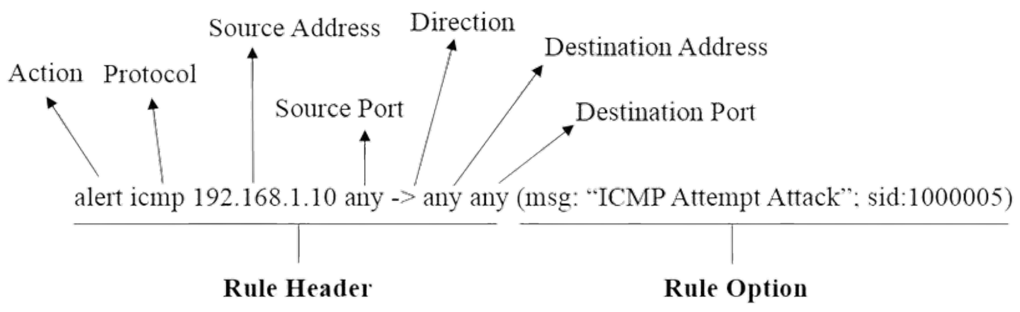
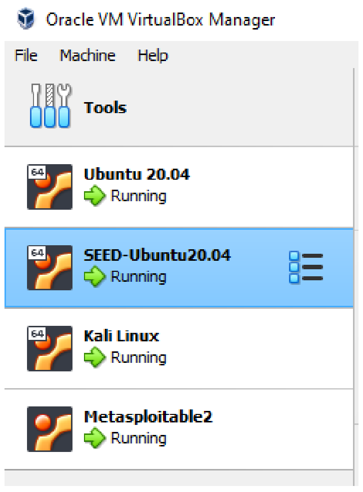


Figure 1: Snort Rule Syntax

2 Prerequisites

2.1 Virtual Machine Setup

Here is the list of Virtual Machines that we are going to use:



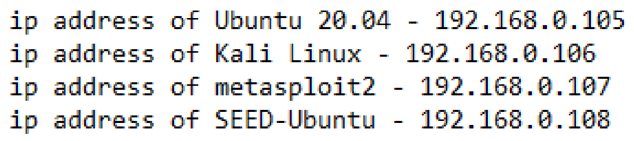
Ubuntu 20.04
Running
SEED-Ubuntu20.04
Running
Kali Linux
Running
Metasploitable2
Running

Figure 2: List of the VMs

The roles of all these virtual machines are:

- **Ubuntu 20.04:** This is where we will install Snort
- **Kali Linux:** Attacker Virtual Machine
- **SEED-Ubuntu 20.04:** A vulnerable Victim Virtual Machine
- **Metasploitable:** Another more vulnerable Victim Virtual Machine

The IP addresses of all of the Virtual Machines must be in promiscuous mode to connect with each other.



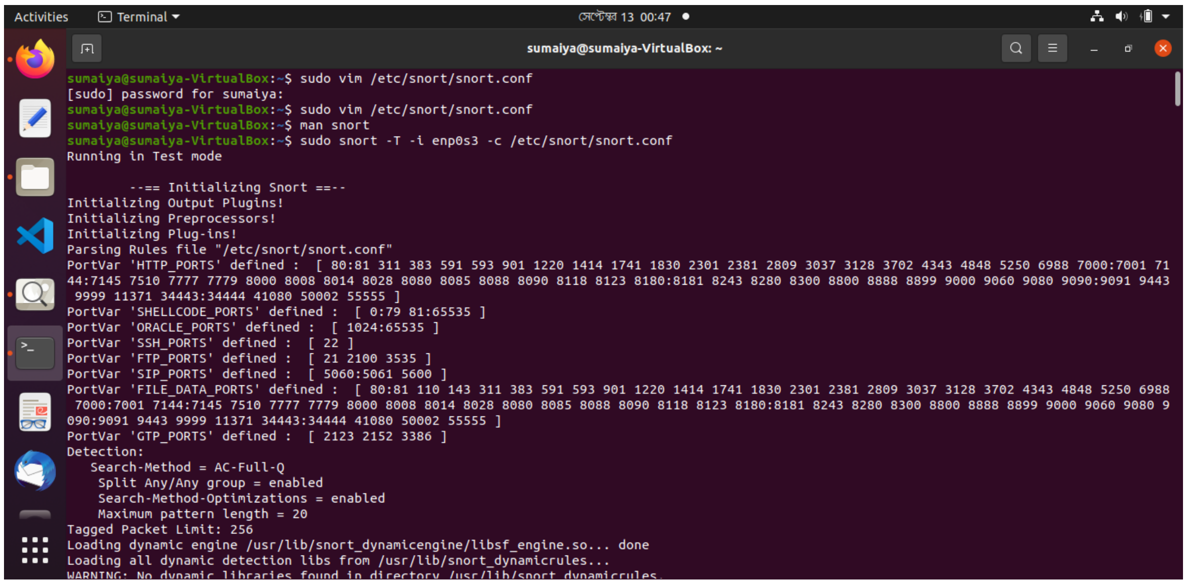
```
ip address of Ubuntu 20.04 - 192.168.0.105
ip address of Kali Linux - 192.168.0.106
ip address of metasploit2 - 192.168.0.107
ip address of SEED-Ubuntu - 192.168.0.108
```

Figure 3: IP Addresses

2.2 Snort Installation

To install Snort on Ubuntu 22.04, we can simply run **”sudo apt-get install -y snort”**. We have installed Snort 2.0 instead of Snort 3.0 because it is more stable and has rich resources available.

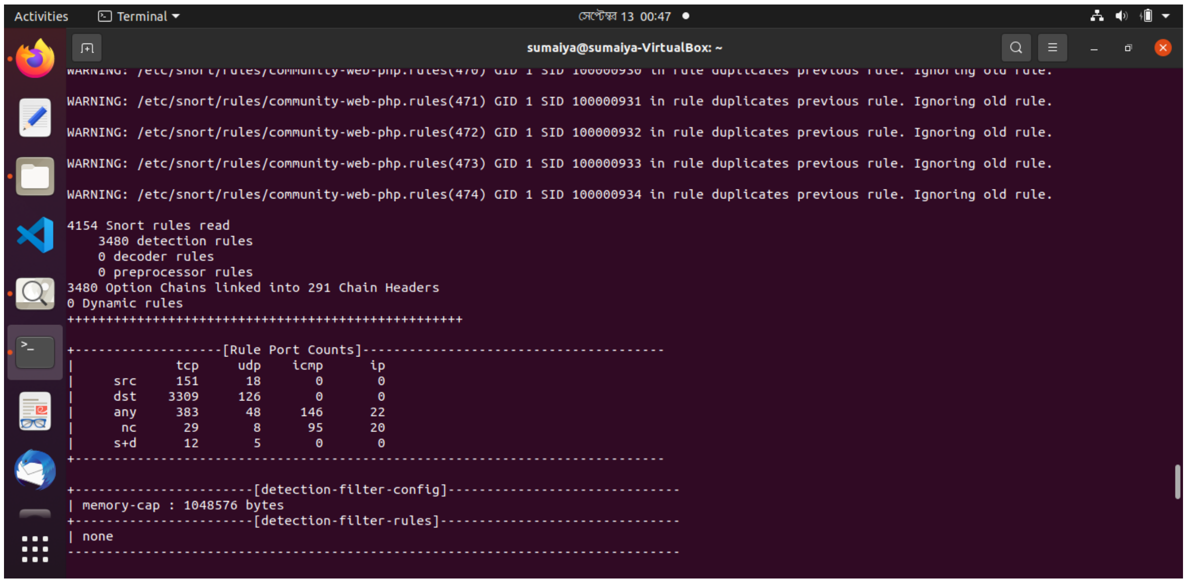
Then we can test the Snort installation to ensure it’s functioning correctly, which is crucial for effective intrusion detection and prevention.



```
sumaiya@sumaiya-VirtualBox:~$ sudo vim /etc/snort/snort.conf
[sudo] password for sumaiya:
sumaiya@sumaiya-VirtualBox:~$ sudo vim /etc/snort/snort.conf
sumaiya@sumaiya-VirtualBox:~$ man snort
sumaiya@sumaiya-VirtualBox:~$ sudo snort -T -i enp0s3 -c /etc/snort/snort.conf
Running in Test mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 71
44:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443
 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988
 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9
090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
Detection:
   Search-Method = AC-Full-Q
    Split Any/Any group = enabled
    Search-Method-Optimizations = enabled
    Maximum pattern length = 20
Tagged Packet Limit: 256
Loading dynamic engine /usr/lib/snort_dynamicengine/libsf_engine.so... done
Loading all dynamic detection libs from /usr/lib/snort_dynamicrules...
WARNING: No dynamic libraries found in directory /usr/lib/snort_dynamicrules.
```

Figure 4: Testing Snort

The image below displays the total number of rules installed in Snort, highlighting the extensive rule set that helps identify and respond to various network threats.



```
WARNING: /etc/snort/rules/community-web-php.rules(470) GID 1 SID 100000930 in rule duplicates previous rule. Ignoring old rule.

WARNING: /etc/snort/rules/community-web-php.rules(471) GID 1 SID 100000931 in rule duplicates previous rule. Ignoring old rule.

WARNING: /etc/snort/rules/community-web-php.rules(472) GID 1 SID 100000932 in rule duplicates previous rule. Ignoring old rule.

WARNING: /etc/snort/rules/community-web-php.rules(473) GID 1 SID 100000933 in rule duplicates previous rule. Ignoring old rule.

WARNING: /etc/snort/rules/community-web-php.rules(474) GID 1 SID 100000934 in rule duplicates previous rule. Ignoring old rule.

4154 Snort rules read
    3480 detection rules
    0 decoder rules
    0 preprocessor rules
3480 Option Chains linked into 291 Chain Headers
0 Dynamic rules
+++++++++++++++++++++++++++++++++++++++++++++++++++

+-------------------[Rule Port Counts]---------------------------------------
|             tcp     udp    icmp      ip
|     src     151      18       0       0
|     dst    3309     126       0       0
|     any     383      48     146      22
|      nc      29       8      95      20
|     s+d      12       5       0       0
+----------------------------------------------------------------------------

+-----------------------[detection-filter-config]------------------------------
| memory-cap : 1048576 bytes
+-----------------------[detection-filter-rules]-------------------------------
| none
-------------------------------------------------------------------------------
```

Figure 5: Total Rules Installed

Inside the **etc/snort/** folder:

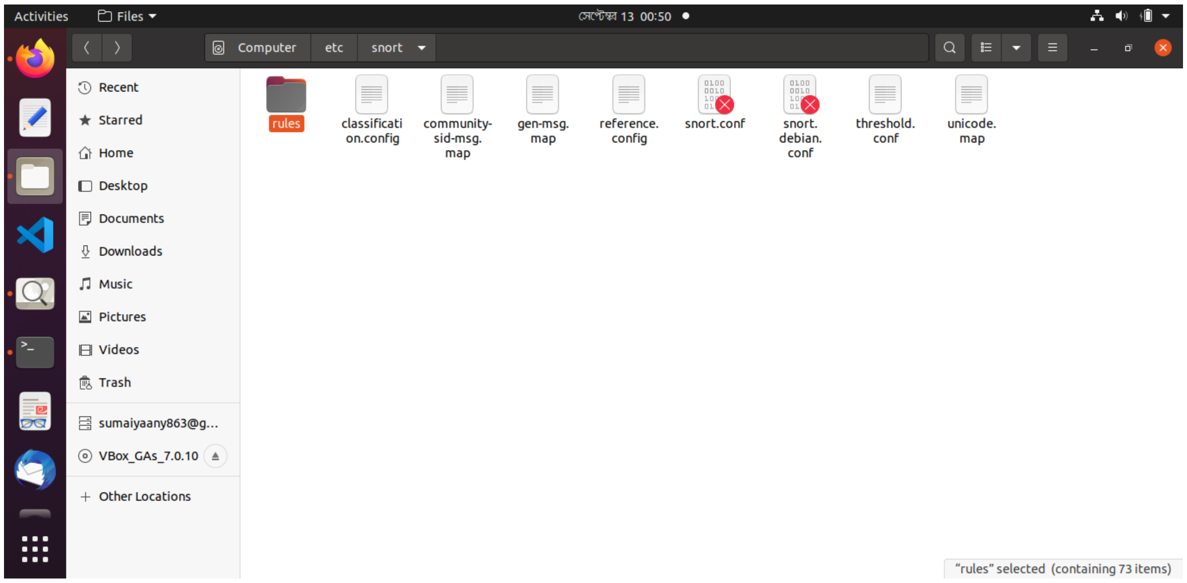
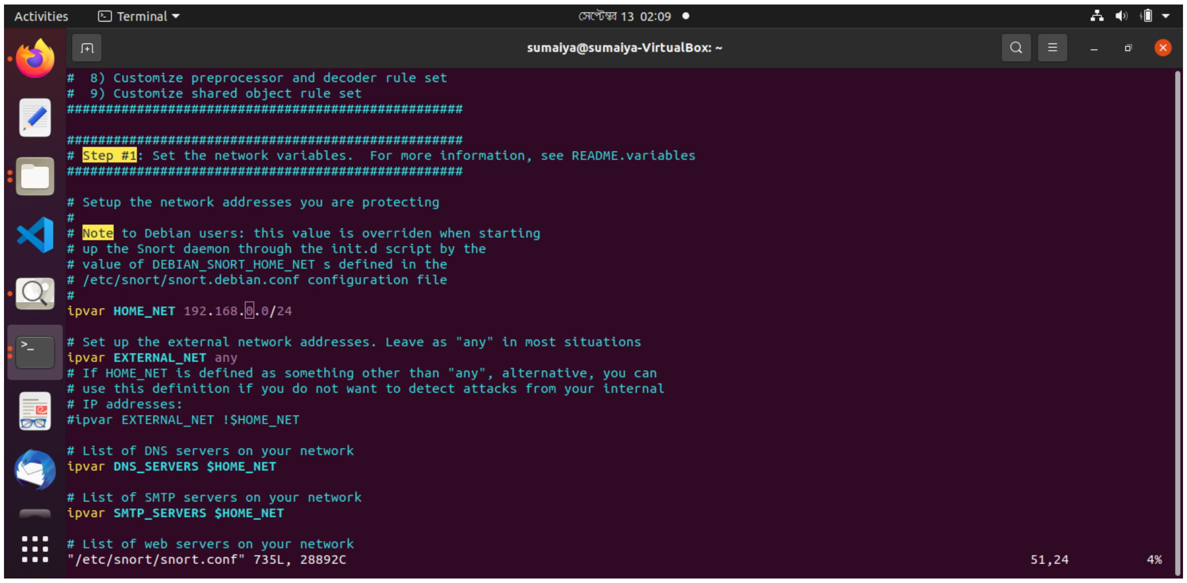


Figure 6: Snort Folder

The image below provides a glimpse into the configuration file **etc/snort/snort.conf**, where Snort's settings are defined:



```
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
###################################################

###################################################
# Step #1: Set the network variables.  For more information, see README.variables
###################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.0.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
"/etc/snort/snort.conf" 735L, 28892C                                    51,24          4%
```

Figure 7: Configuration File

The Rules folder contains a collection of rule sets that Snort uses to detect known attack patterns:

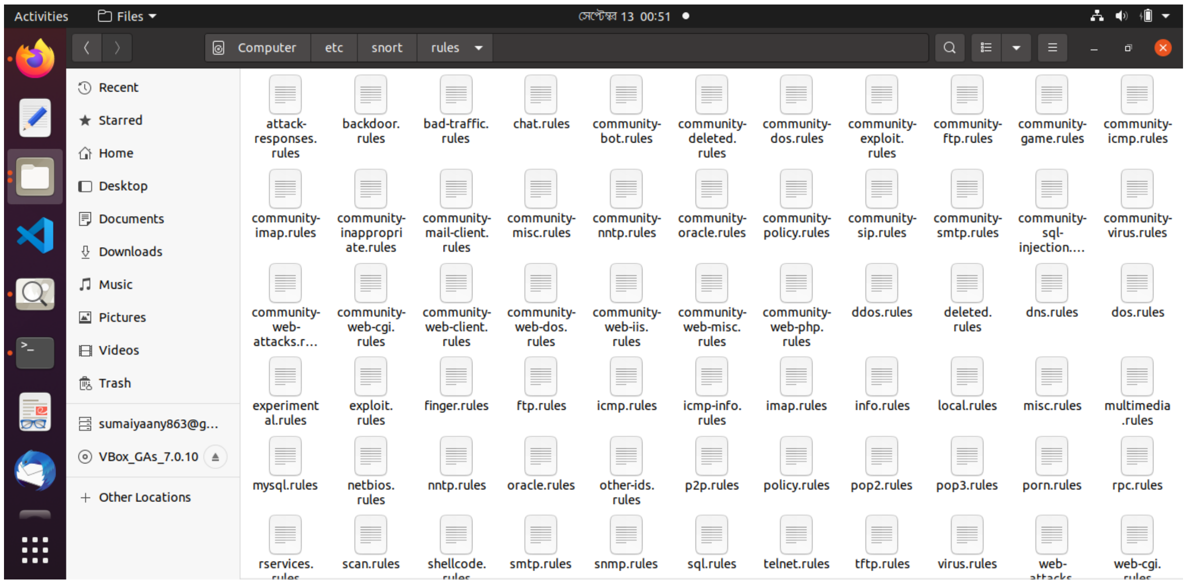
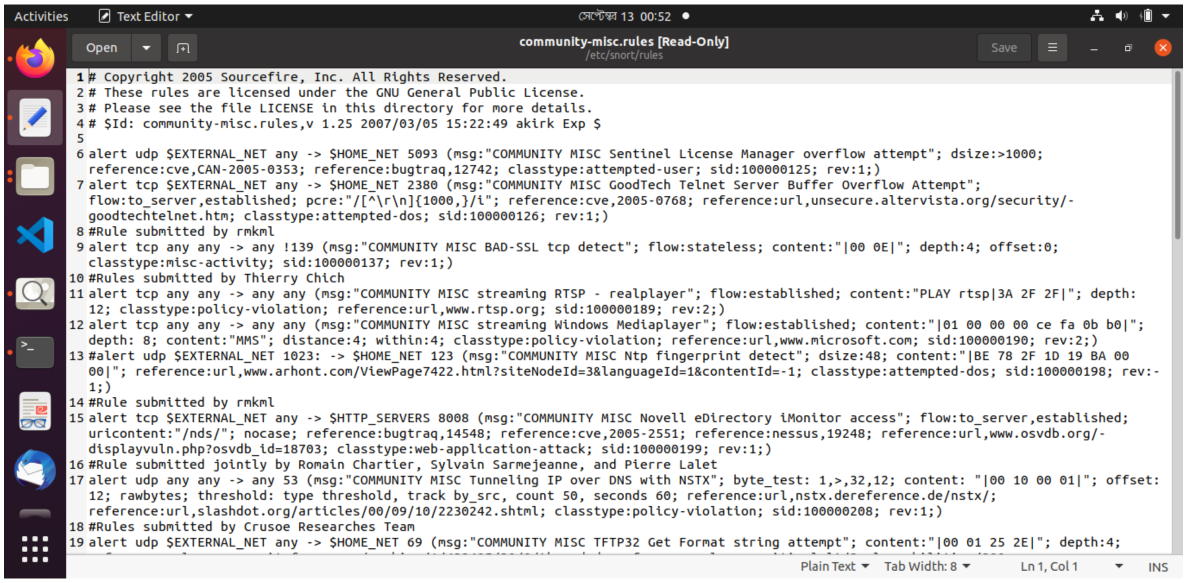


Figure 8: Rules Folder

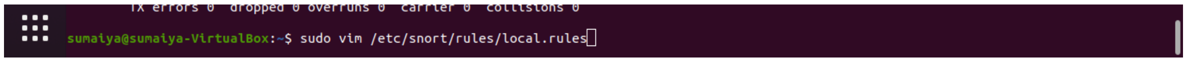
Here, you can see the contents of the Community Rules File which is free for everybody:



```
# Copyright 2005 Sourcefire, Inc. All Rights Reserved.
# These rules are licensed under the GNU General Public License.
# Please see the file LICENSE in this directory for more details.
# $Id: community-misc.rules,v 1.25 2007/03/05 15:22:49 akirk Exp $

alert udp $EXTERNAL_NET any -> $HOME_NET 5093 (msg:"COMMUNITY MISC Sentinel License Manager overflow attempt"; dsize:>1000;
reference:cve,CAN-2005-0353; reference:bugtraq,12742; classtype:attempted-user; sid:100000125; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 2380 (msg:"COMMUNITY MISC GoodTech Telnet Server Buffer Overflow Attempt";
flow:to_server,established; pcre:"/[^\r\n]{1000,}/i"; reference:cve,2005-0768; reference:url,unsecure.altervista.org/security/-
goodtechtelnet.htm; classtype:attempted-dos; sid:100000126; rev:1;)
#Rule submitted by rmkml
alert tcp any any -> any !139 (msg:"COMMUNITY MISC BAD-SSL tcp detect"; flow:stateless; content:"|00 0E|"; depth:4; offset:0;
classtype:misc-activity; sid:100000137; rev:1;)
#Rules submitted by Thierry Chich
alert tcp any any -> any any (msg:"COMMUNITY MISC streaming RTSP - realplayer"; flow:established; content:"PLAY rtsp|3A 2F 2F|"; depth:
12; classtype:policy-violation; reference:url,www.rtsp.org; sid:100000189; rev:2;)
alert tcp any any -> any any (msg:"COMMUNITY MISC streaming Windows Mediaplayer"; flow:established; content:"|01 00 00 00 ce fa 0b b0|";
depth: 8; content:"MMS"; distance:4; within:4; classtype:policy-violation; reference:url,www.microsoft.com; sid:100000190; rev:2;)
#alert udp $EXTERNAL_NET 1023: -> $HOME_NET 123 (msg:"COMMUNITY MISC Ntp fingerprint detect"; dsize:48; content:"|BE 78 2F 1D 19 BA 00
00|"; reference:url,www.arhont.com/ViewPage7422.html?siteNodeId=3&languageId=1&contentId=-1; classtype:attempted-dos; sid:100000198; rev:-
1;)
#Rule submitted by rmkml
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 8008 (msg:"COMMUNITY MISC Novell eDirectory iMonitor access"; flow:to_server,established;
uricontent:"/nds/"; nocase; reference:bugtraq,14548; reference:cve,2005-2551; reference:nessus,19248; reference:url,www.osvdb.org/-
displayvuln.php?osvdb_id=18703; classtype:web-application-attack; sid:100000199; rev:1;)
#Rule submitted jointly by Romain Chartier, Sylvain Sarmejeanne, and Pierre Lalet
alert udp any any -> any 53 (msg:"COMMUNITY MISC Tunneling IP over DNS with NSTX"; byte_test: 1,>,32,12; content: "|00 10 00 01|"; offset:
12; rawbytes; threshold: type threshold, track by_src, count 50, seconds 60; reference:url,nstx.dereference.de/nstx/;
reference:url,slashdot.org/articles/00/09/10/2230242.shtml; classtype:policy-violation; sid:100000208; rev:1;)
#Rules submitted by Crusoe Researches Team
alert udp $EXTERNAL_NET any -> $HOME_NET 69 (msg:"COMMUNITY MISC TFTP32 Get Format string attempt"; content:"|00 01 25 2E|"; depth:4;
```

Figure 9: Community Rules File

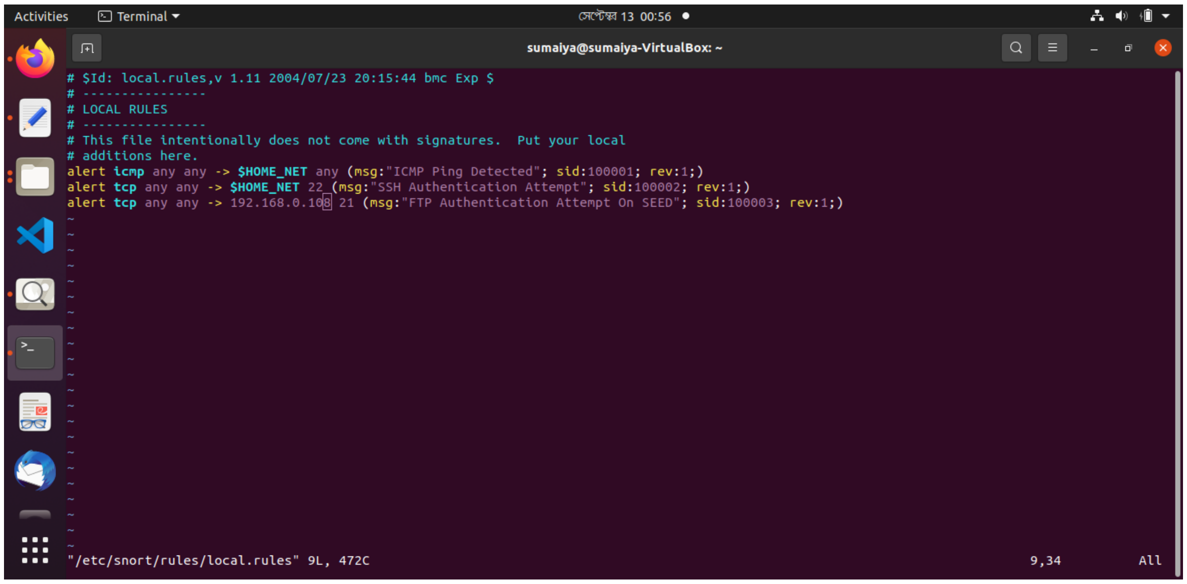
The command to see local rules:



```
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
sumaiya@sumaiya-VirtualBox:~$ sudo vim /etc/snort/rules/local.rules
```

Figure 10: Command for Local Rules

Inside Local Rules File:



```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected"; sid:100001; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Attempt"; sid:100002; rev:1;)
alert tcp any any -> 192.168.0.108 21 (msg:"FTP Authentication Attempt On SEED"; sid:100003; rev:1;)
"/etc/snort/rules/local.rules" 9L, 472C                                    9,34          All
```

Figure 11: Local rules file

3 Demonstration

3.1 Packet Sniffing

Packet sniffing in Snort refers to the capability of the Snort Intrusion Detection System (IDS) to capture and inspect network packets as they traverse a network interface. It is one of the fundamental functions of Snort, allowing it to analyze network traffic in real-time for the detection of suspicious or malicious activity.

Packet sniffing specifically refers to the process of capturing and inspecting network packets as they traverse a network interface. This is the initial step where the IDS monitors the raw network traffic for any suspicious or malicious activity. So, we are not demonstrating this feature separately and we are jumping to Intrusion Detection directly.

3.2 Network Intrusion Detection

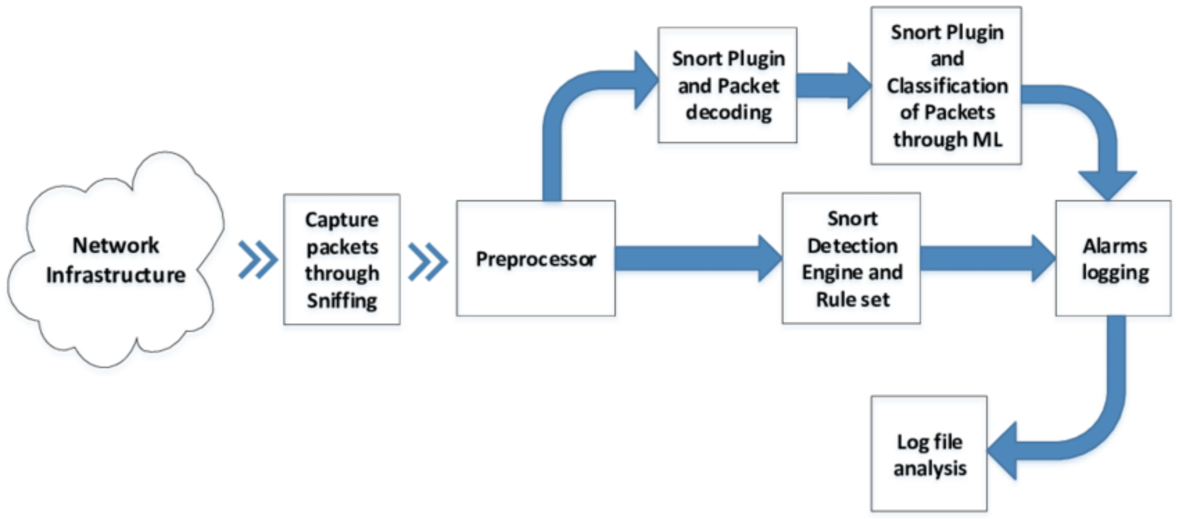

Network Infrastructure
Capture packets through Sniffing
Preprocessor
Snort Plugin and Packet decoding
Snort Plugin and Classification of Packets through ML
Snort Detection Engine and Rule set
Alarms logging
Log file analysis

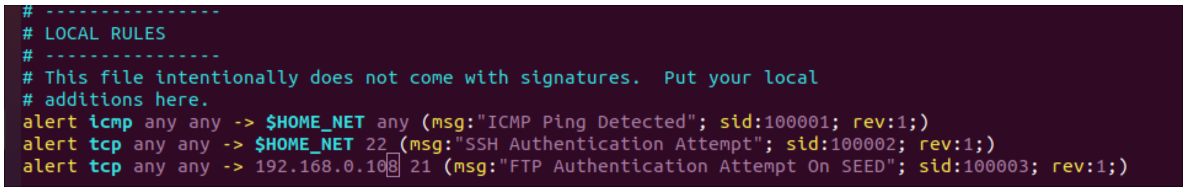
Figure 12: IDS Workflow

Intrusion detection is the process of actively discovering threats/attacks/intrusions on a network, different hosts, or services. An Intrusion Detection System (IDS) is a system/host planted within a network to capture traffic and identify malicious activity based on predefined rules, after which, this malicious activity is logged, and a notification is sent to the relevant parties informing them of an intrusion.
There are 2 types of IDS solutions based on the placement:

- **HIDS:** A Host-based IDS (HIDS) is set up on an individual host on a network.
- **NIDS:** A Network IDS (NIDS) is placed within a network to monitor traffic to and from all hosts on a network.

Intrusion detection systems are typically coupled with the functionality to also perform intrusion prevention, whereby specific rules can be set to drop packets that are malicious or intrusive.

We first write some rules for our demonstration:



```
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected"; sid:100001; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Attempt"; sid:100002; rev:1;)
alert tcp any any -> 192.168.0.108 21 (msg:"FTP Authentication Attempt On SEED"; sid:100003; rev:1;)
```

Figure 13: IDS Rules

Here we use a tool named **Snorpy** to write the rules easily:

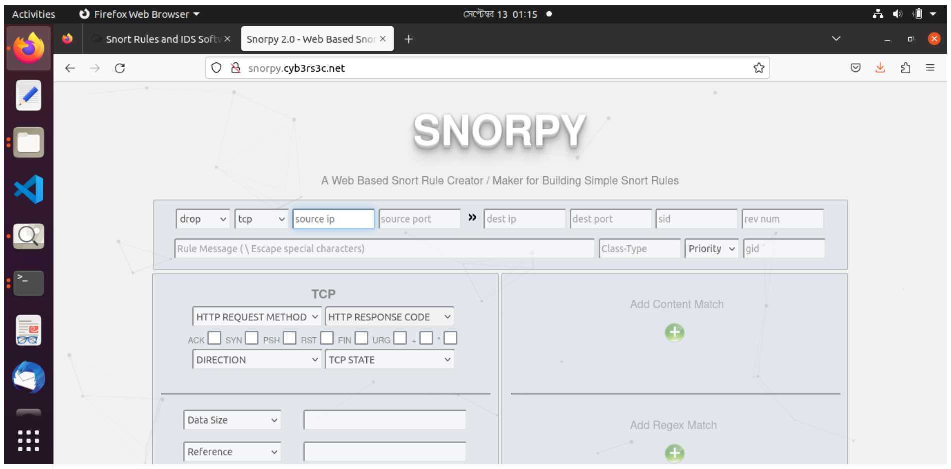
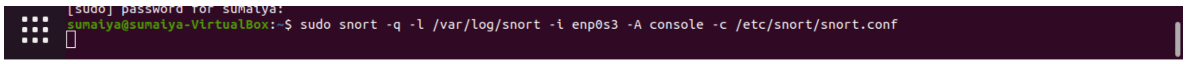


Figure 14: Snorpy

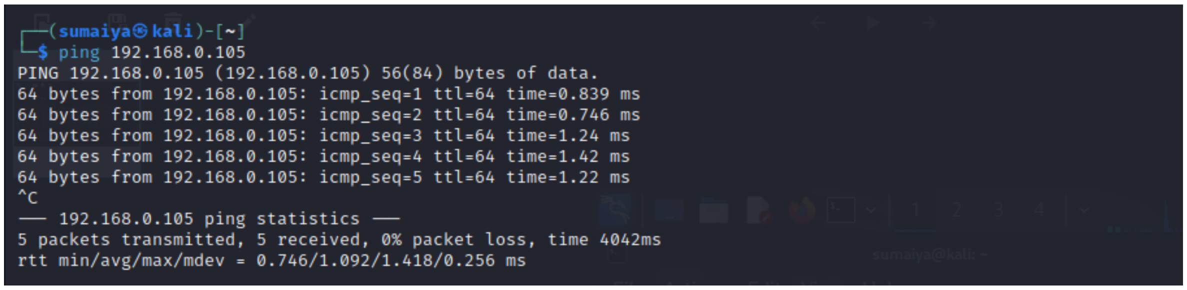
This screenshot displays the command used to start the Snort Intrusion Detection System in our Ubuntu 22.04 VM:



```
[sudo] password for sumaiya:
sumaiya@sumaiya-VirtualBox:~$ sudo snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf
```

Figure 15: Command to Run Snort

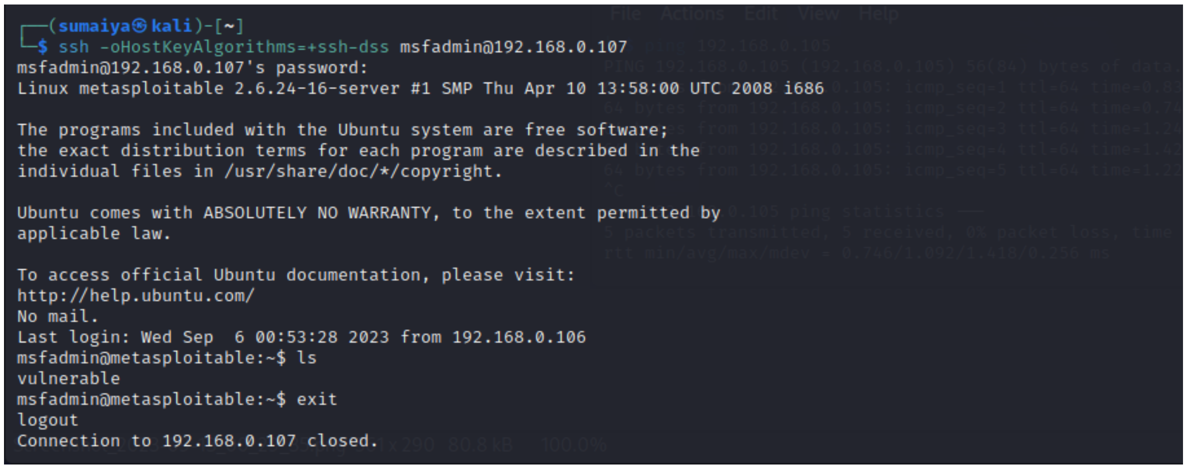
A ping command is being executed from Kali Linux to test the connectivity to our Ubuntu 20.04. Ping tests help ensure network connectivity and are often used as part of network troubleshooting processes.



```
┌──(sumaiya㉿kali)-[~]
└─$ ping 192.168.0.105
PING 192.168.0.105 (192.168.0.105) 56(84) bytes of data.
64 bytes from 192.168.0.105: icmp_seq=1 ttl=64 time=0.839 ms
64 bytes from 192.168.0.105: icmp_seq=2 ttl=64 time=0.746 ms
64 bytes from 192.168.0.105: icmp_seq=3 ttl=64 time=1.24 ms
64 bytes from 192.168.0.105: icmp_seq=4 ttl=64 time=1.42 ms
64 bytes from 192.168.0.105: icmp_seq=5 ttl=64 time=1.22 ms
^C
--- 192.168.0.105 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4042ms
rtt min/avg/max/mdev = 0.746/1.092/1.418/0.256 ms
```

Figure 16: Pinging Ubuntu 20.04

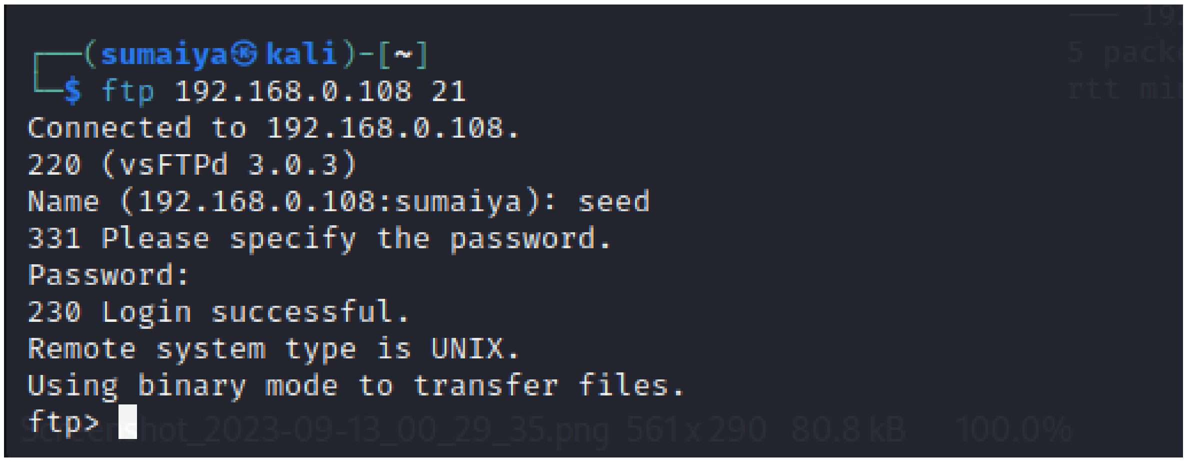
To establish an SSH connection to Metasploit2 with specific configuration options:



```
┌──(sumaiya㉿kali)-[~]
└─$ ssh -oHostKeyAlgorithms=+ssh-dss msfadmin@192.168.0.107
msfadmin@192.168.0.107's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Wed Sep  6 00:53:28 2023 from 192.168.0.106
msfadmin@metasploitable:~$ ls
vulnerable
msfadmin@metasploitable:~$ exit
logout
Connection to 192.168.0.107 closed.
```

Figure 17: SSH to Metasploit2

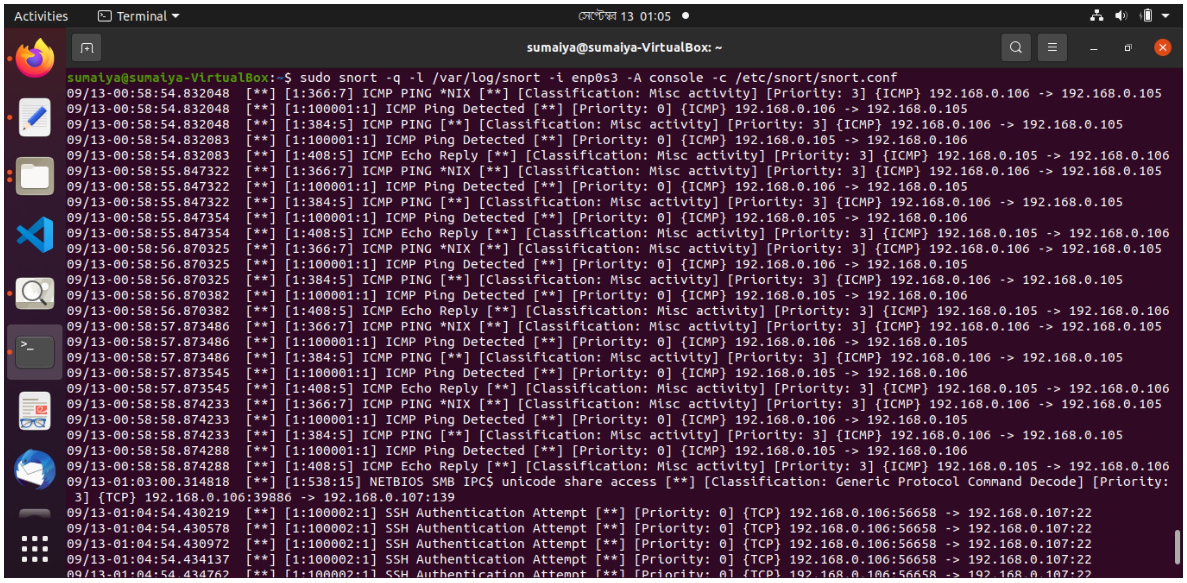
To initiate an FTP (File Transfer Protocol) connection to our SEED VM on port 21:



```
┌──(sumaiya㉿kali)-[~]
└─$ ftp 192.168.0.108 21
Connected to 192.168.0.108.
220 (vsFTPd 3.0.3)
Name (192.168.0.108:sumaiya): seed
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

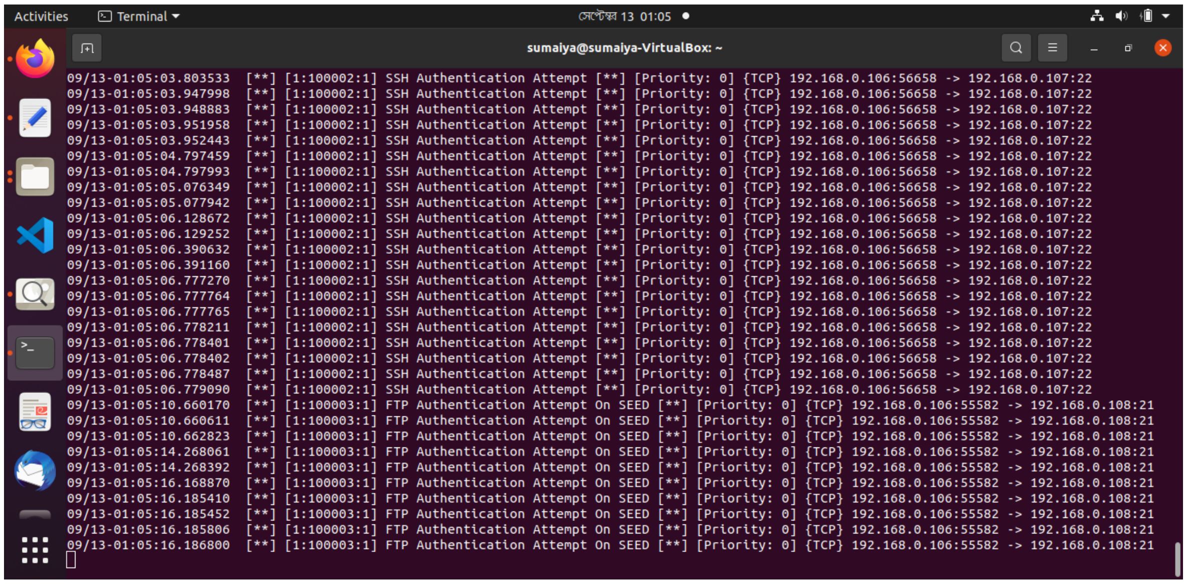
Figure 18: FTP to SEED

Ensuring the rules are correctly configured:



```
sumaiya@sumaiya-VirtualBox:~$ sudo snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf
09/13-00:58:54.832048  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:54.832048  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:54.832048  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:54.832083  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:54.832083  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:55.847322  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:55.847322  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:55.847322  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:55.847354  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:55.847354  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:56.870325  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:56.870325  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:56.870325  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:56.870382  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:56.870382  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:57.873486  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:57.873486  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:57.873486  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:57.873545  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:57.873545  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:58.874233  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:58.874233  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:58.874233  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.106 -> 192.168.0.105
09/13-00:58:58.874288  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-00:58:58.874288  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.105 -> 192.168.0.106
09/13-01:03:00.314818  [**] [1:538:15] NETBIOS SMB IPC$ unicode share access [**] [Classification: Generic Protocol Command Decode] [Priority:
 3] {TCP} 192.168.0.106:39886 -> 192.168.0.107:139
09/13-01:04:54.430219  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:04:54.430578  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:04:54.430972  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:04:54.434137  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:04:54.434762  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
```

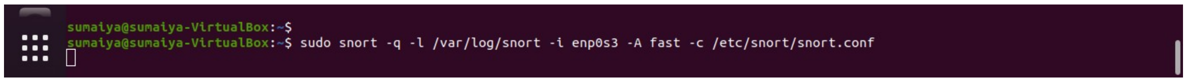
Figure 19: Verifying Local Rules (1)



```
09/13-01:05:03.803533  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:03.947998  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:03.948883  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:03.951958  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:03.952443  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:04.797459  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:04.797993  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:05.076349  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:05.077942  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.128672  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.129252  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.390632  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.391160  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.777270  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.777764  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.777765  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.778211  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.778401  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.778402  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.778487  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:06.779090  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.0.106:56658 -> 192.168.0.107:22
09/13-01:05:10.660170  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:10.660611  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:10.662823  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:14.268061  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:14.268392  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:16.168870  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:16.185410  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:16.185452  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:16.185806  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
09/13-01:05:16.186800  [**] [1:100003:1] FTP Authentication Attempt On SEED [**] [Priority: 0] {TCP} 192.168.0.106:55582 -> 192.168.0.108:21
```

Figure 20: Verifying Local Rules (2)

Now we check the log for any intrusion:



```
sumaiya@sumaiya-VirtualBox:~$
sumaiya@sumaiya-VirtualBox:~$ sudo snort -q -l /var/log/snort -i enp0s3 -A fast -c /etc/snort/snort.conf
```

Figure 21: Command to Log Alerts

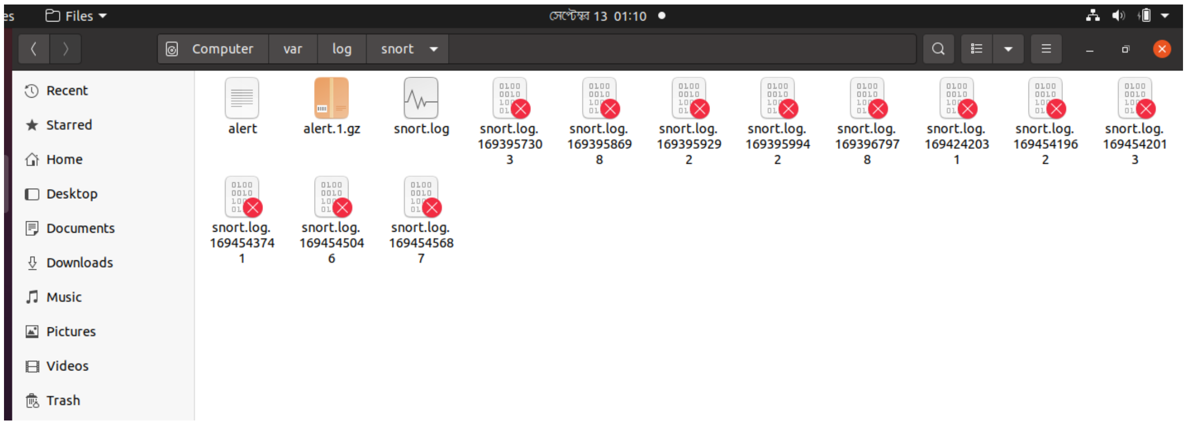
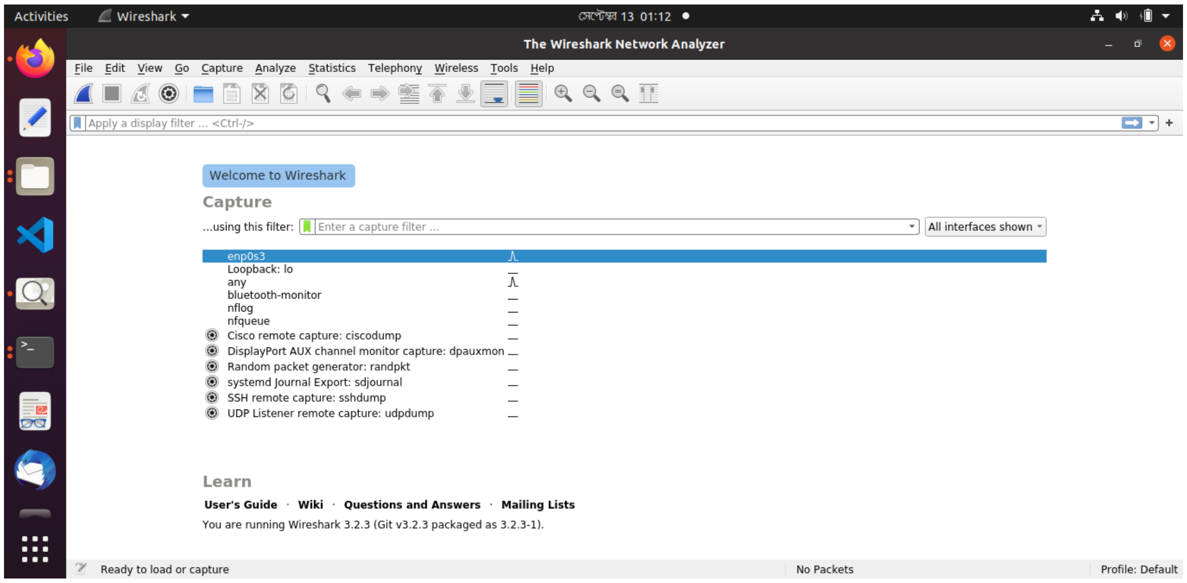


Figure 22: Snort Log Folder

We can see the alert has been generated by Snort after detecting an intrusion. All the logs are saved too.

3.3 Packet Logging

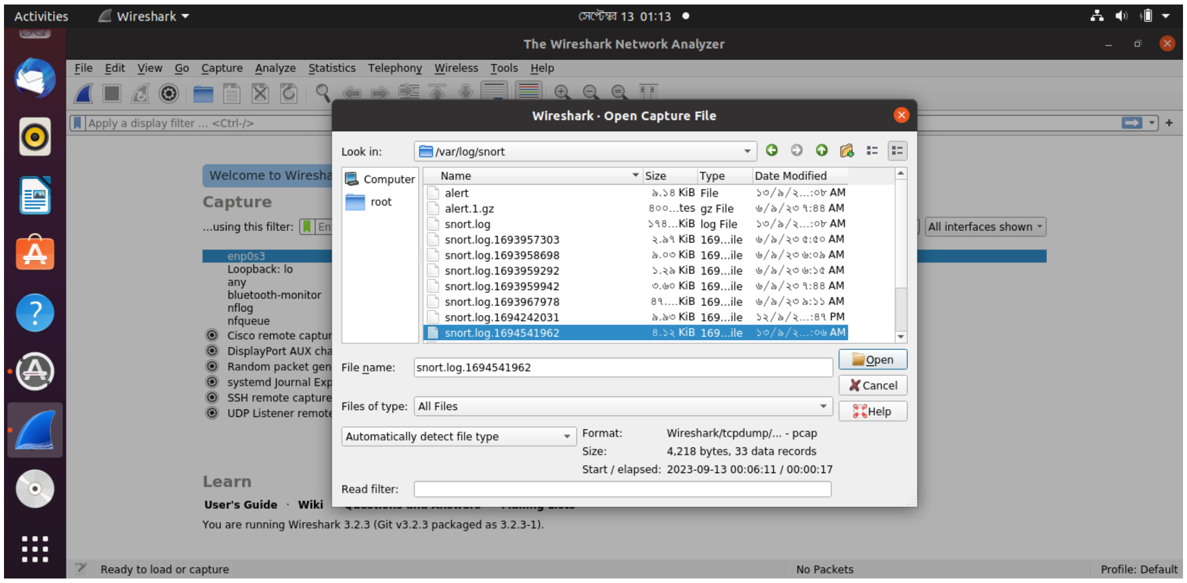
Packet logging in Snort refers to the process of capturing and recording network packets that match specific rules or signatures defined in the Snort intrusion detection system (IDS) configuration. Here we will use a tool named **Wireshark** for logging.


Activities
Wireshark
সেপ্টেম্বর 13 01:12
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
enp0s3
Loopback: lo
any
bluetooth-monitor
nflog
nfqueue
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
Ready to load or capture
No Packets
Profile: Default

You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).

Figure 23: Wireshark

We open the last log file generated:



Wireshark · Open Capture File

Look in: /var/log/snort

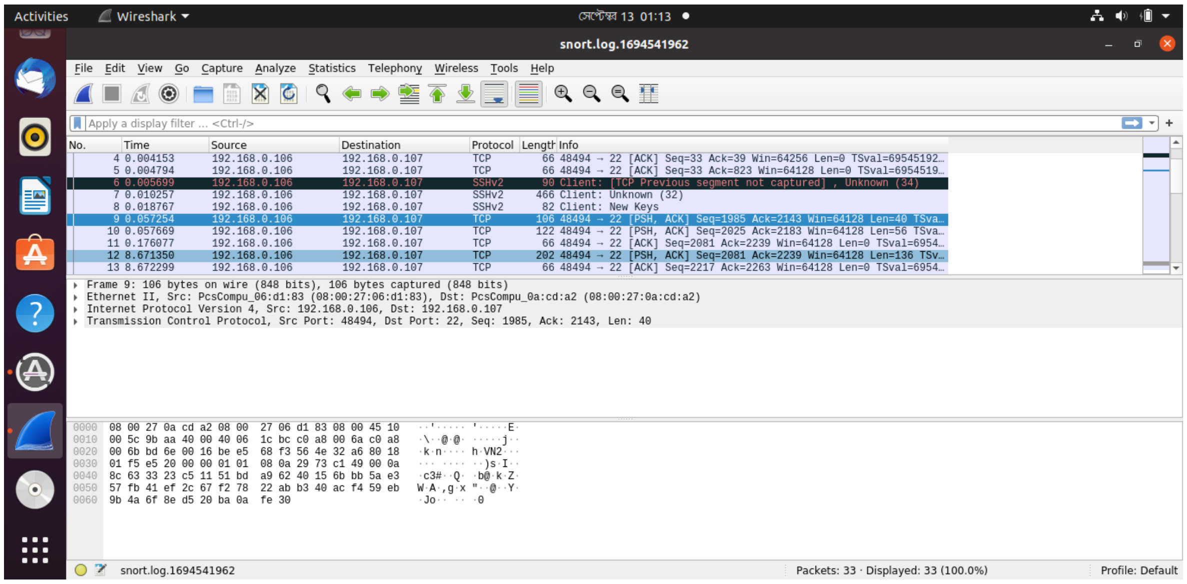
Name	Size	Type	Date Modified
alert	৯.১৪ KiB	File	১৩/৯/২...:০৮ AM
alert.1.gz	৪০০...tes	gz File	৬/৯/২৩ ৭:৪৪ AM
snort.log	১৭৪...KiB	log File	১৩/৯/২...:০৮ AM
snort.log.1693957303	২.৯৭ KiB	169...ile	৬/৯/২৩ ৫:৫০ AM
snort.log.1693958698	৯.০৩ KiB	169...ile	৬/৯/২৩ ৬:০৯ AM
snort.log.1693959292	১.২৯ KiB	169...ile	৬/৯/২৩ ৬:১৫ AM
snort.log.1693959942	৩.৬০ KiB	169...ile	৬/৯/২৩ ৭:৪৪ AM
snort.log.1693967978	৪৭....KiB	169...ile	৬/৯/২৩ ৯:১১ AM
snort.log.1694242031	৯.৯৩ KiB	169...ile	১২/৯/২...:৪৭ PM
snort.log.1694541962	৪.১২ KiB	169...ile	১৩/৯/২...:০৬ AM

File name: snort.log.1694541962
Files of type: All Files
Automatically detect file type
Format: Wireshark/tcpdump/... - pcap
Size: 4,218 bytes, 33 data records
Start / elapsed: 2023-09-13 00:06:11 / 00:00:17
Read filter:
Open Cancel Help

You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).

Figure 24: Open Capture File in Wireshark

Here, we can see the detailed information of every alert:


Activities
Wireshark
সেপ্টেম্বর 13 01:13
snort.log.1694541962
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Profile: Default

```
No.  Time      Source         Destination    Protocol Length Info
   4 0.004153  192.168.0.106  192.168.0.107  TCP        66 48494 → 22 [ACK] Seq=33 Ack=39 Win=64256 Len=0 TSval=69545192…
   5 0.004794  192.168.0.106  192.168.0.107  TCP        66 48494 → 22 [ACK] Seq=33 Ack=823 Win=64128 Len=0 TSval=6954519…
   6 0.005699  192.168.0.106  192.168.0.107  SSHv2      90 Client: [TCP Previous segment not captured] , Unknown (34)
   7 0.010257  192.168.0.106  192.168.0.107  SSHv2     466 Client: Unknown (32)
   8 0.018767  192.168.0.106  192.168.0.107  SSHv2      82 Client: New Keys
   9 0.057254  192.168.0.106  192.168.0.107  TCP       106 48494 → 22 [PSH, ACK] Seq=1985 Ack=2143 Win=64128 Len=40 TSva…
  10 0.057669  192.168.0.106  192.168.0.107  TCP       122 48494 → 22 [PSH, ACK] Seq=2025 Ack=2183 Win=64128 Len=56 TSva…
  11 0.176077  192.168.0.106  192.168.0.107  TCP        66 48494 → 22 [ACK] Seq=2081 Ack=2239 Win=64128 Len=0 TSval=6954…
  12 8.671350  192.168.0.106  192.168.0.107  TCP       202 48494 → 22 [PSH, ACK] Seq=2081 Ack=2239 Win=64128 Len=136 TSv…
  13 8.672299  192.168.0.106  192.168.0.107  TCP        66 48494 → 22 [ACK] Seq=2217 Ack=2263 Win=64128 Len=0 TSval=6954…

▸ Frame 9: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)
▸ Ethernet II, Src: PcsCompu_06:d1:83 (08:00:27:06:d1:83), Dst: PcsCompu_0a:cd:a2 (08:00:27:0a:cd:a2)
▸ Internet Protocol Version 4, Src: 192.168.0.106, Dst: 192.168.0.107
▸ Transmission Control Protocol, Src Port: 48494, Dst Port: 22, Seq: 1985, Ack: 2143, Len: 40

0000  08 00 27 0a cd a2 08 00  27 06 d1 83 08 00 45 10   ··'····· '·····E·
0010  00 5c 9b aa 40 00 40 06  1c bc c0 a8 00 6a c0 a8   ·\··@·@· ·····j··
0020  00 6b bd 6e 00 16 be e5  68 f3 56 4e 32 a6 80 18   ·k·n···· h·VN2···
0030  01 f5 e5 20 00 00 01 01  08 0a 29 73 c1 49 00 0a   ···  ··· ··)s·I··
0040  8c 63 33 23 c5 11 51 bd  a9 62 40 15 6b bb 5a e3   ·c3#··Q· ·b@·k·Z·
0050  57 fb 41 ef 2c 67 f2 78  22 ab b3 40 ac f4 59 eb   W·A·,g·x "··@··Y·
0060  9b 4a 6f 8e d5 20 ba 0a  fe 30                     ·Jo·· ·· ·0

snort.log.1694541962          Packets: 33 · Displayed: 33 (100.0%)
```

Figure 25: Logging in Wireshark

Using this information, we can do any kind of analysis we want.

4 Conclusion

In conclusion, Snort is a powerful and versatile intrusion detection and prevention system that plays a crucial role in network security. It operates based on predefined rules and can analyze network traffic in real-time, making it effective in detecting and responding to a wide range of malicious activities. Whether used for packet logging, packet sniffing, network intrusion detection, or network intrusion prevention, Snort provides valuable insights into network traffic and helps organizations defend against evolving cyber threats. Its open-source nature and extensive community support make it a valuable tool in the arsenal of network security solutions, ensuring the continued resilience of modern network infrastructures.

Resources:

- Snort Official Website
- Youtube Playlist on Snort By HackerSploit