Final Project of CSE406 (Computer Security Sessional)

Project : Snort 3
Tool : NIDS

Submission Date: March 9, 2024

submitted by

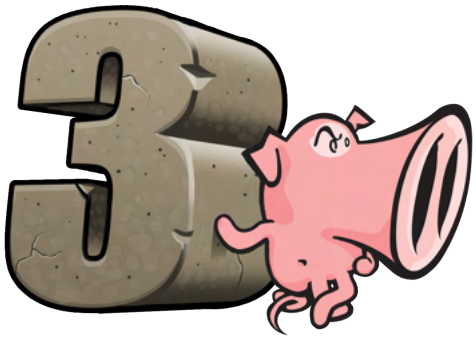
1905048 (Al-Amin Sany)
1905052 (Bijoy Ahmed Saiem)

Under the supervision of

Abdur Rashid Tushar
Lecturer
Computer Science and Engineering Department
Bangladesh University of Engineering and Technology

Contents

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.

2 Introduction

2.1 What is Snort3

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.

2.2 Snort Operations

- **Packet Sniffing:** Analyzes the actual network traffic in real-time
- **Network Intrusion Detection and Prevention:** Analyzes packets and matches traffic against signatures and drop packets.
- **Packet Logging:** Collects and logs network traffic into a log file

2.3 How Does Snort3 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).

2.4 Snort3 Rules

- **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.

The syntax of customized rules are:

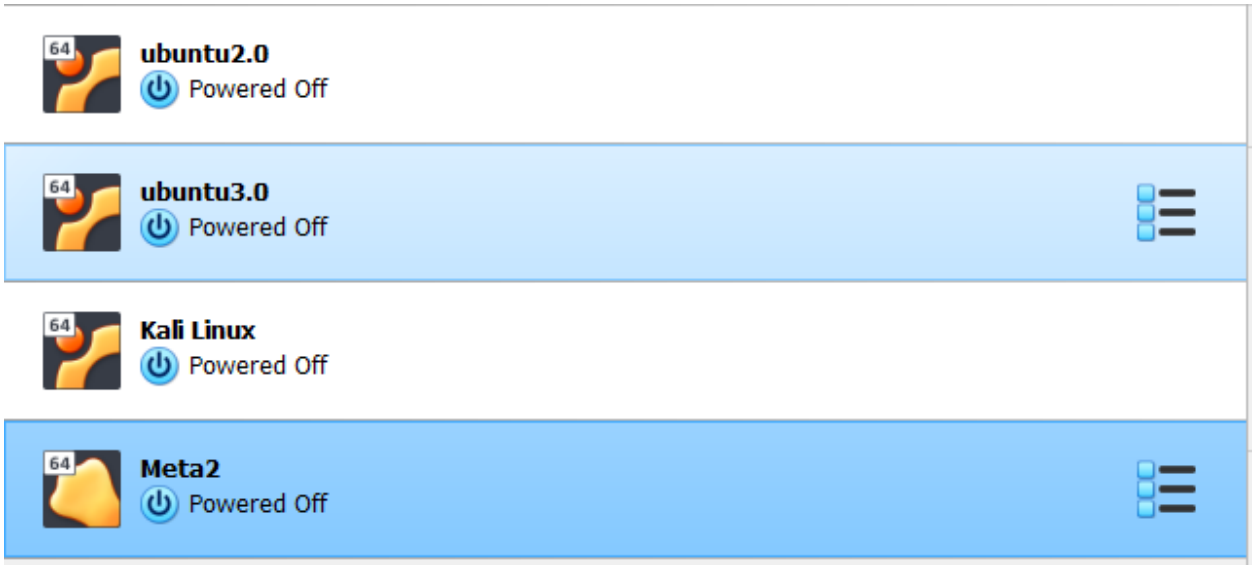
Rule Description:

alert	Action
icmp	Protocol
any	Source Address
any	Source Port
->	Direction
$HOME_NET	Destination Address
any	Destination Port
msg: "Incoming ICMP"	Message / Description
sid:1000001	Rule ID Number
rev:1	Rule Revision Number

3 Prerequisites

3.1 Virtual Machine Setup

Four VMs are used for this project:



The roles of these VMs are:

- **ubuntu2.0:** Snort3 is installed here.
- **ubuntu3.0:** Vulnerable Machine.
- **kali linux:** Attacker Machine
- **Meta2:** More vulnerable Machine.

The IP addresses of all Virtual Machines must be within the same subnet to connect with each other.

IP addresses of the four VMs:

- **ubuntu2.0:** 192.168.48.4
- **ubuntu3.0:** 192.168.48.5
- **kali linux:** 192.168.48.6
- **Meta2:** 192.168.48.7

3.2 Snort3 Installation

We follow this link to install snort3:
https://cytoolz.com/blog/snort-3-install-and-configure-intrusion-detection-system-on-ubuntu-22-04

Youtube Link:
https://youtu.be/uPdCmuFh40M?si=gJXhh7eJ9ibMkw4R

4 Demonstration

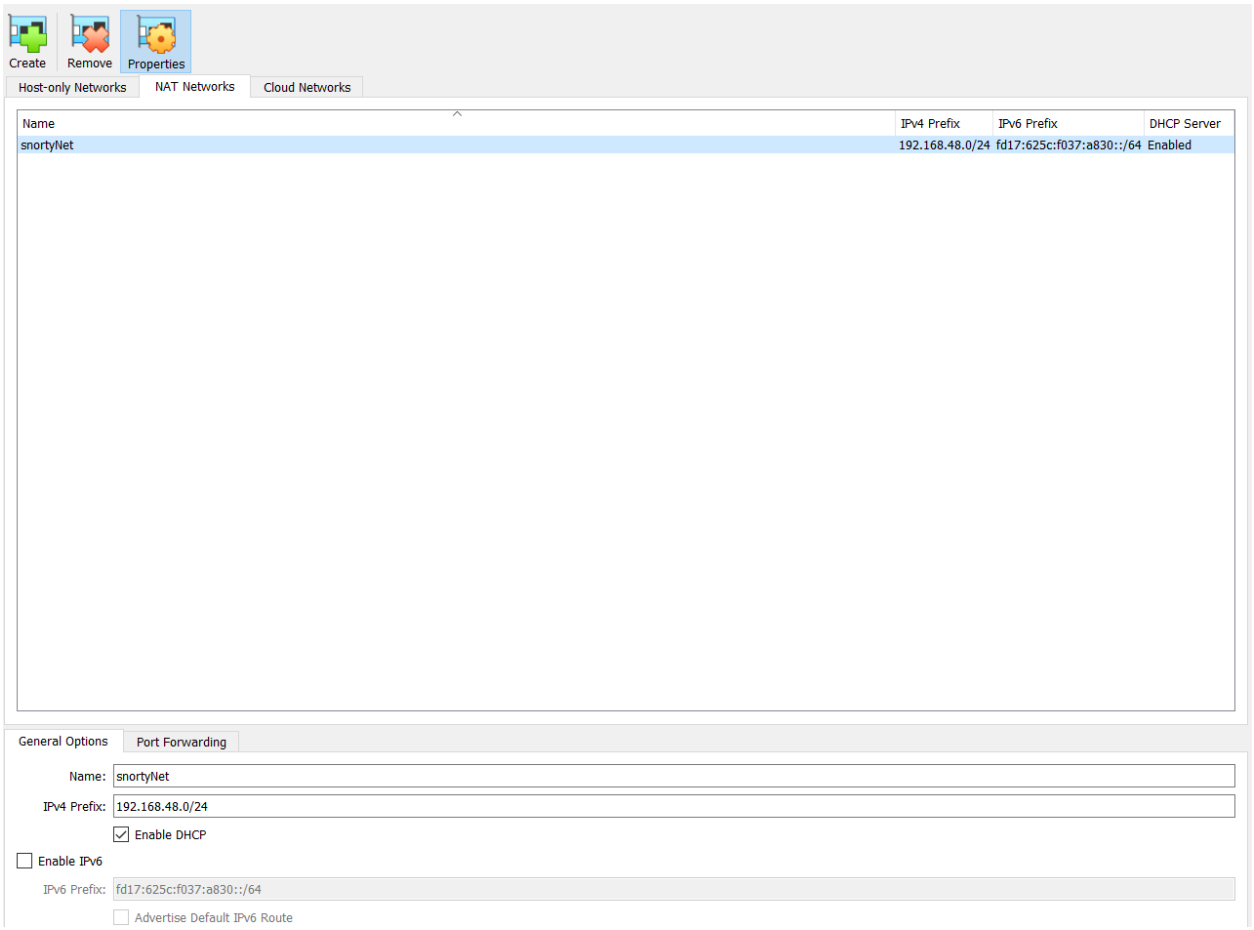
4.1 Pakcet Sniffing

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.

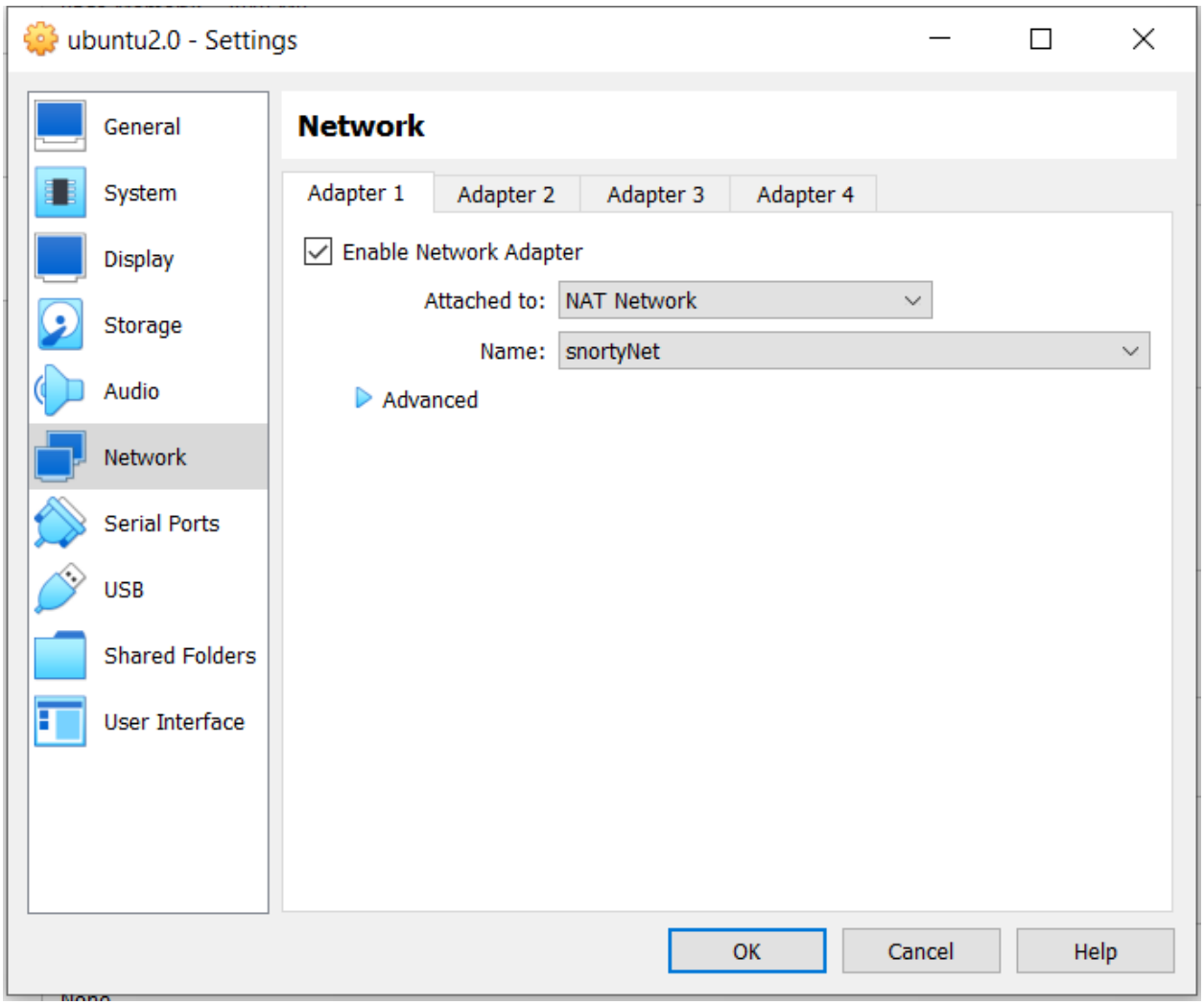
4.2 Network Intrution Detection

Snort3 rules operate similarly to conventional firewall rules. They are designed to analyze network activity, looking for specific patterns or signatures. When a match is found, Snort3 can take action by either generating an alert or dropping the packet if configured as an Intrusion Prevention System (IPS). This proactive approach helps in enhancing network security by identifying and responding to potential threats in real-time. For verifying this a home network is created below.

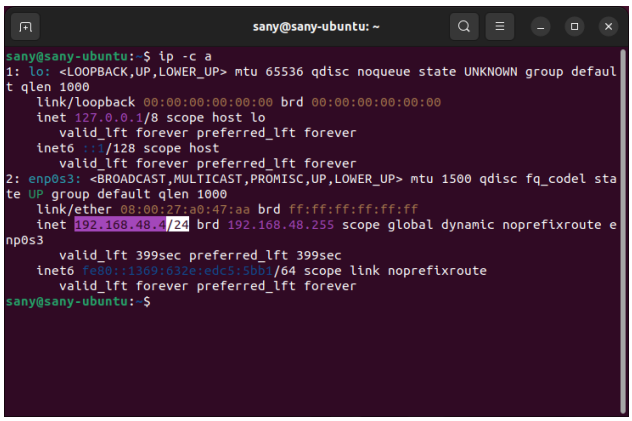
- **A Home Network(Name : snortyNet , IP : 192.168.48.0/24) is Created:**

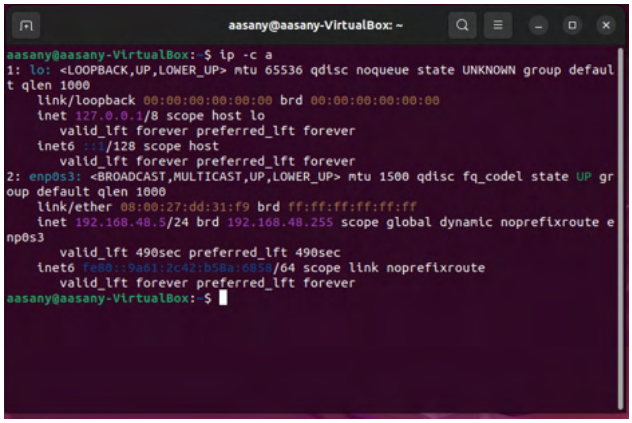


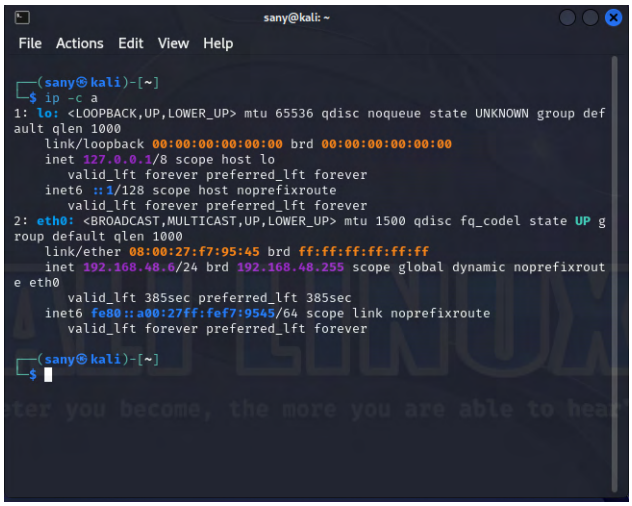
- **Virtual Machines are connected to the Home Network:**

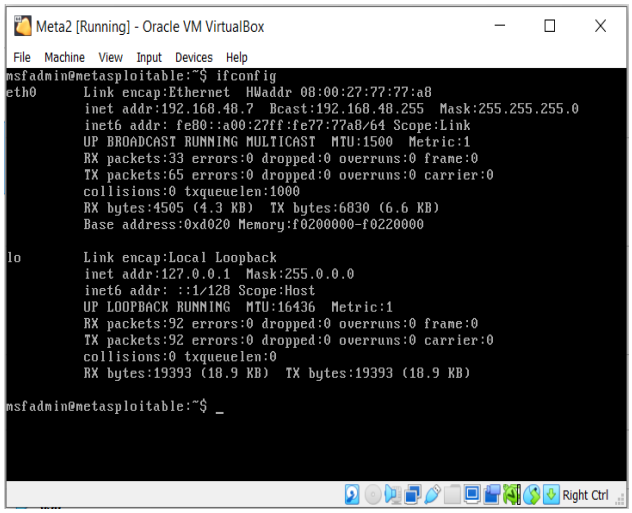


- The IP addresses of all Virtual Machines are set within the same subnet to connect with each other.



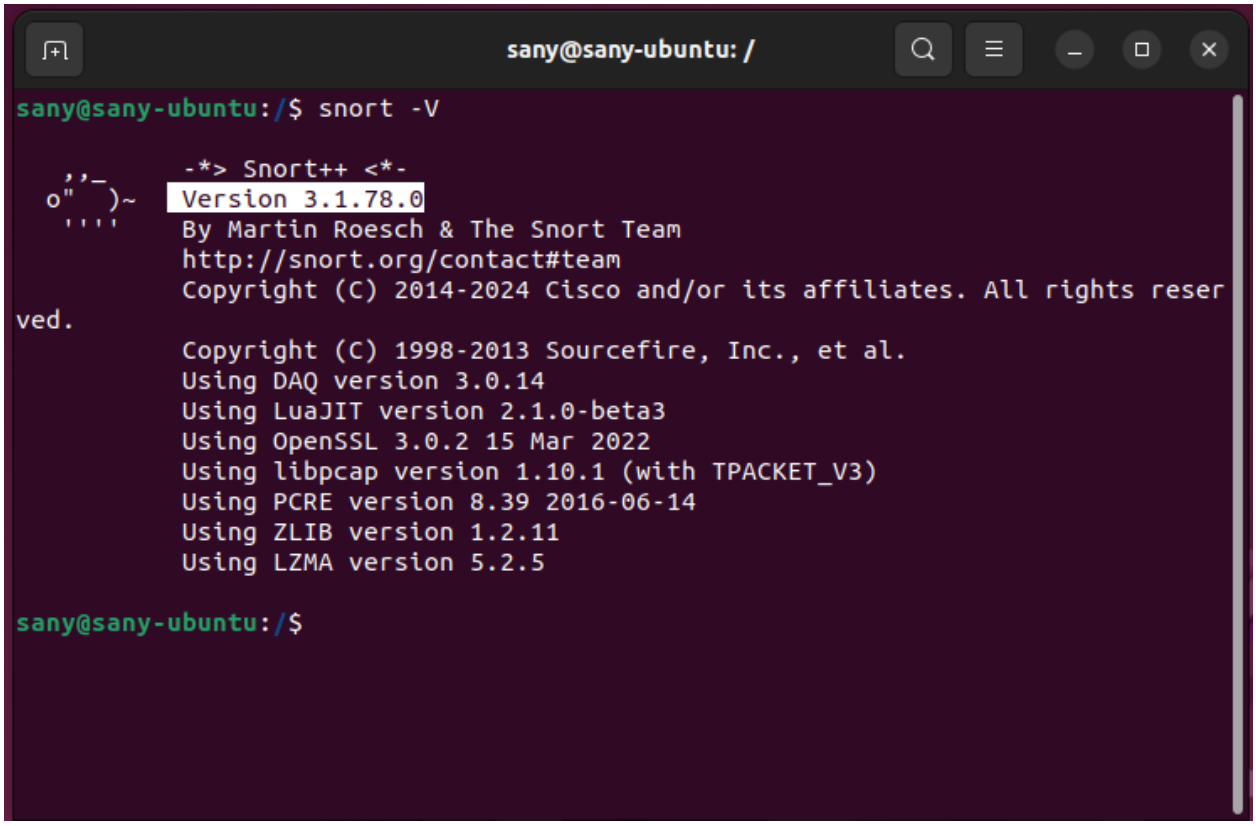






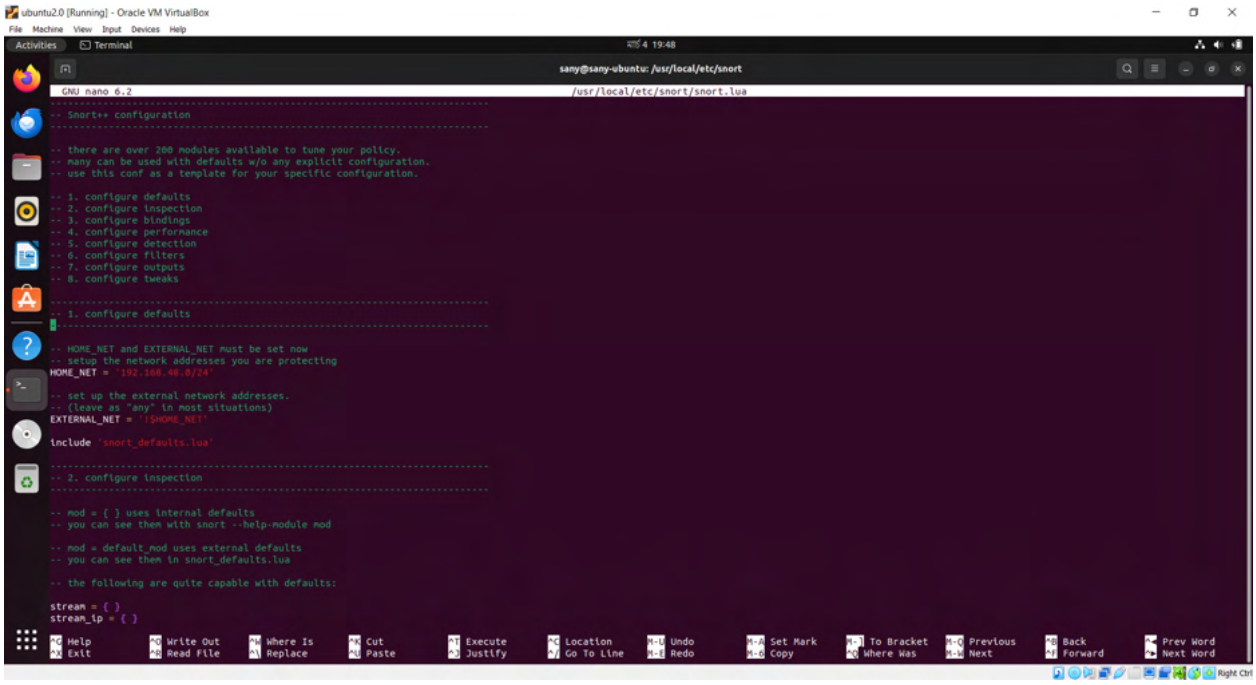
- **Threading and shared memory allow us to scale Snort3 to our network and create a much faster start-up.**
- **This allows multiple packet processing to free up more memory for more packet processing power.**

So, we use snort3 instead of snort2. The version of our snort is:



Snort3 Configuration

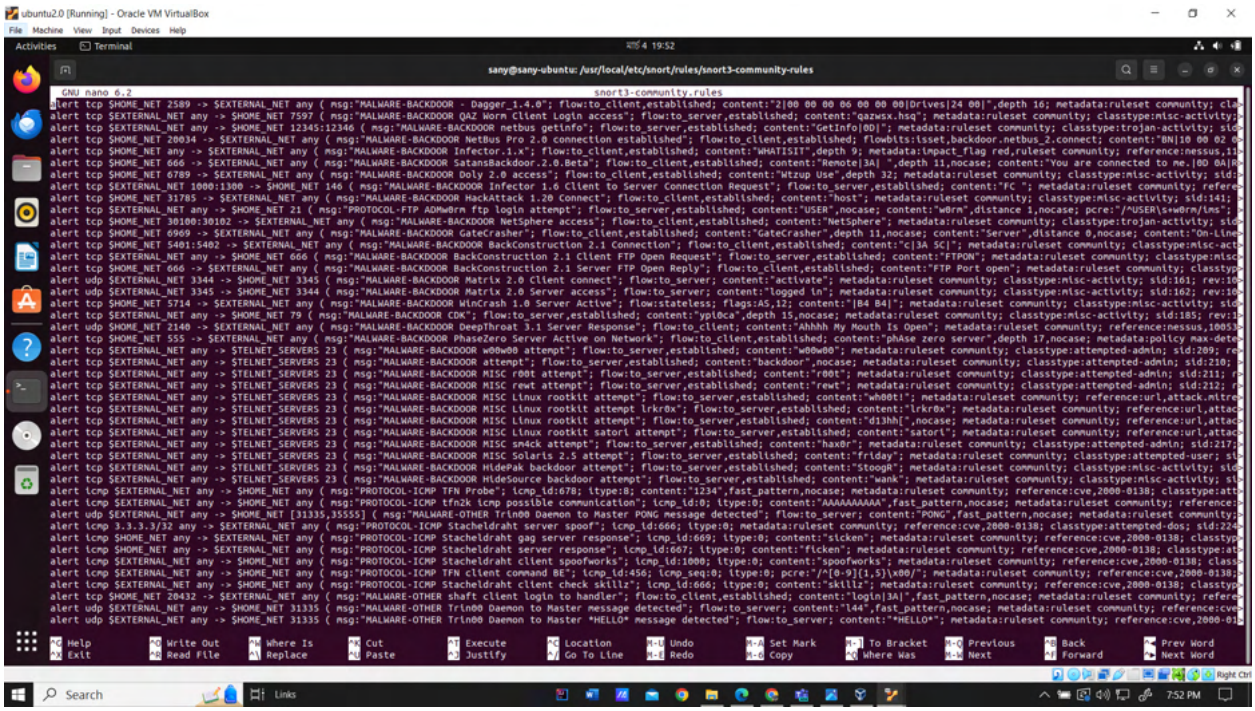
- Home NET:192.168.48.0/24
- EXTERNEL NET:!HOME NET



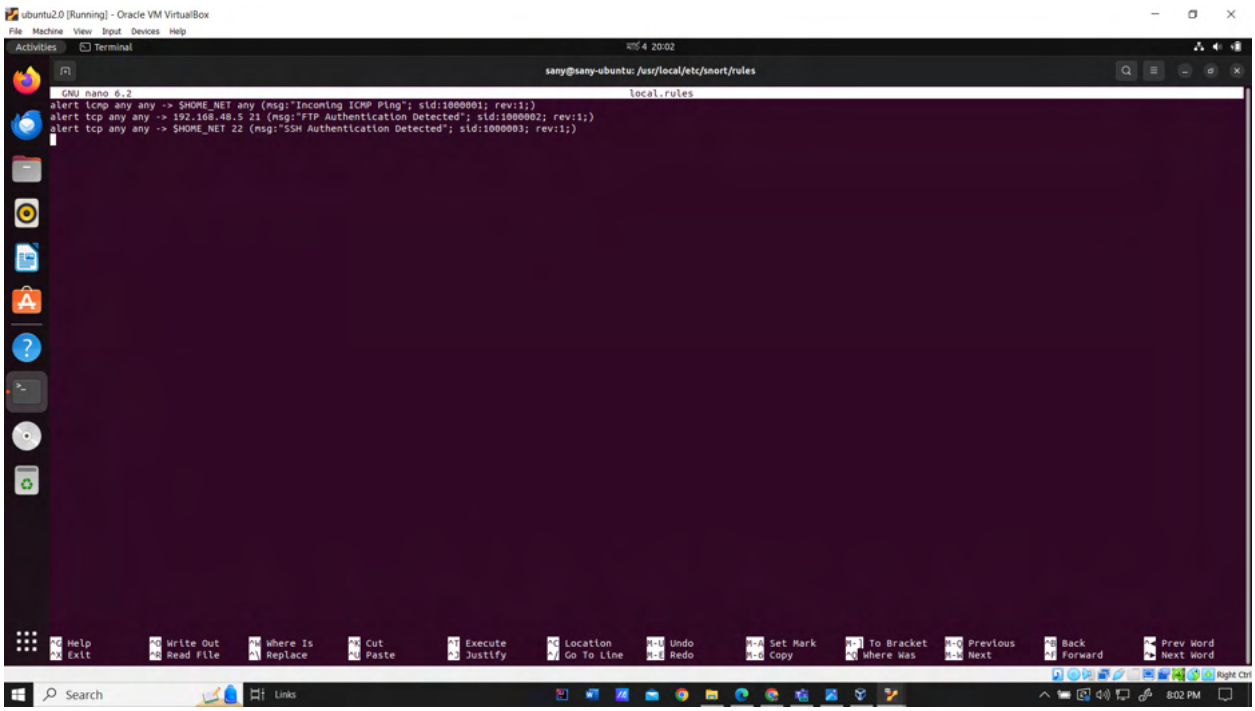
- **local.rules file created to write custom rules:**

```
ips =
{
    -- use this to enable decoder and inspector alerts
    --enable_builtin_rules = true,

    -- use include for rules files; be sure to set your path
    -- note that rules files can include other rules files
    -- (see also related path vars at the top of snort_defaults.lua)

    variables = default_variables,
    rules = [[
      include /usr/local/etc/snort/rules/local.rules
      include /usr/local/etc/snort/rules/snort3-community-rules/snort3-community.rules
    ]]
}
```

Snort3 Community Rules



Snort3 Custom Rules



Custom rules explanation

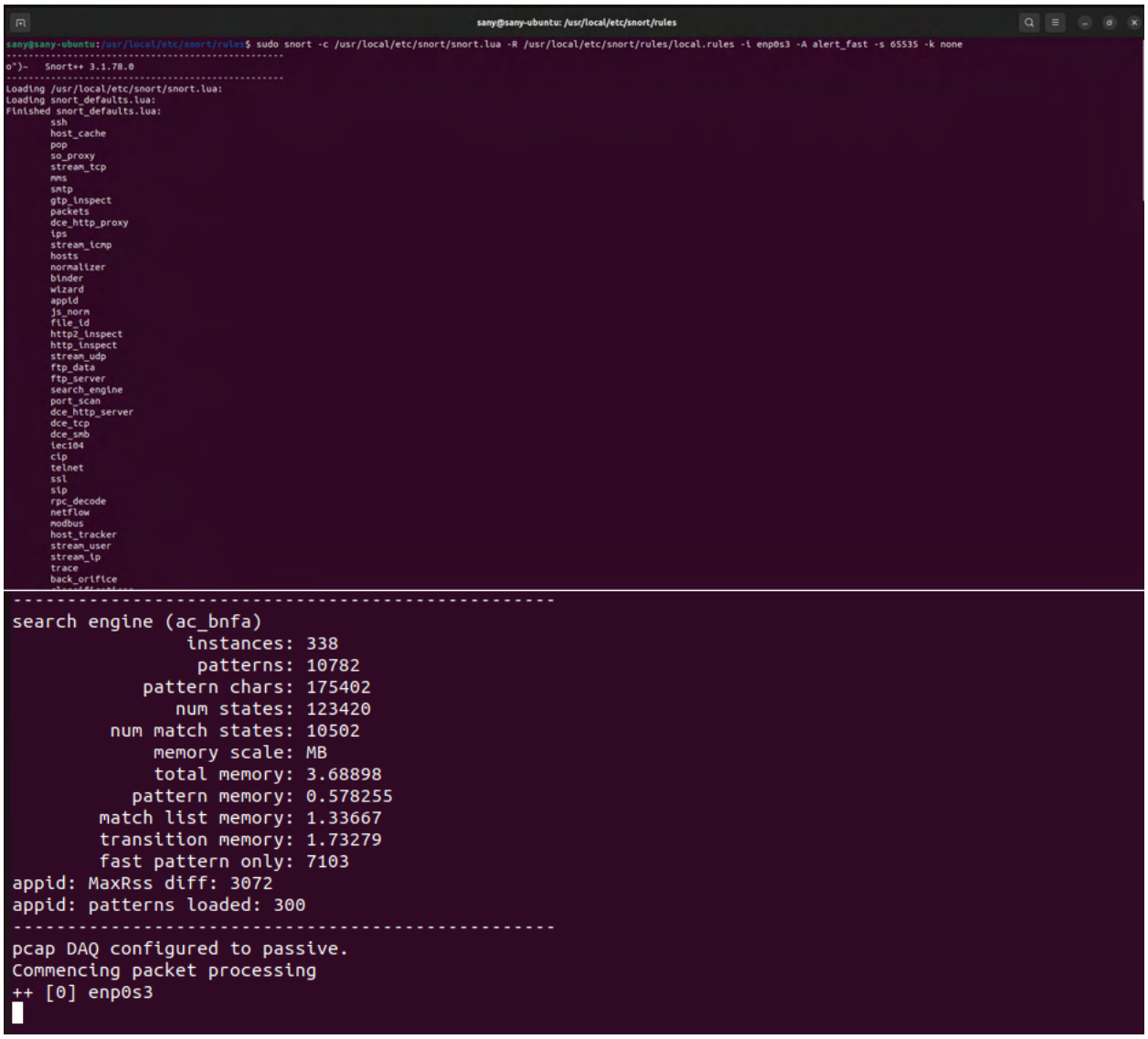
Rule Description:

alert	Action
icmp	Protocol
any	Source Address
any	Source Port
->	Direction
$HOME_NET	Destination Address
any	Destination Port
msg: "Incoming ICMP"	Message / Description
sid:1000001	Rule ID Number
rev:1	Rule Revision Number

Snort3 manual trigger

Command description:

-c */usr/local/etc/snort/snort.lua*	The `-c` option specifies the configuration file to be used.
-R */usr/local/etc/snort/rules/local.rules*	The `-R` option specifies the rule file to be used.
-i *ens160*	The `-i` option specifies the network interface to listen on.
-A *alert_fast*	The `-A` option sets the alert mode. The 'alert_fast' mode set to prioritize performance over extensive logging
-s *65535*	The `-s` option sets the snaplen and it's set to 65535, equivalent to the entire packet to capture.
-k *none*	The `-k` option sets the checksum mode. Setting to 'none' indicates that Snort deactivates the IP/TCP/UDP checksum validation.


sany@sany-ubuntu: /usr/local/etc/snort/rules
sany@sany-ubuntu:/usr/local/etc/snort/rules$ sudo snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/local.rules -i enp0s3 -A alert_fast -s 65535 -k none
o")~ Snort++ 3.1.78.0
Loading /usr/local/etc/snort/snort.lua:
Loading snort_defaults.lua:
Finished snort_defaults.lua:
ssh
host_cache
pop
so_proxy
stream_tcp
mms
smtp
gtp_inspect
packets
dce_http_proxy
ips
stream_icmp
hosts
normalizer
binder
wizard
appid
js_norm
file_id
http2_inspect
http_inspect
stream_udp
ftp_data
ftp_server
search_engine
port_scan
dce_http_server
dce_tcp
dce_smb
iec104
cip
telnet
ssl
sip
rpc_decode
netflow
modbus
host_tracker
stream_user
stream_ip
trace
back_orifice
search engine (ac_bnfa)
instances: 338
patterns: 10782
pattern chars: 175402
num states: 123420
num match states: 10502
memory scale: MB
total memory: 3.68898
pattern memory: 0.578255
match list memory: 1.33667
transition memory: 1.73279
fast pattern only: 7103
appid: MaxRss diff: 3072
appid: patterns loaded: 300
pcap DAQ configured to passive.
Commencing packet processing
++ [0] enp0s3

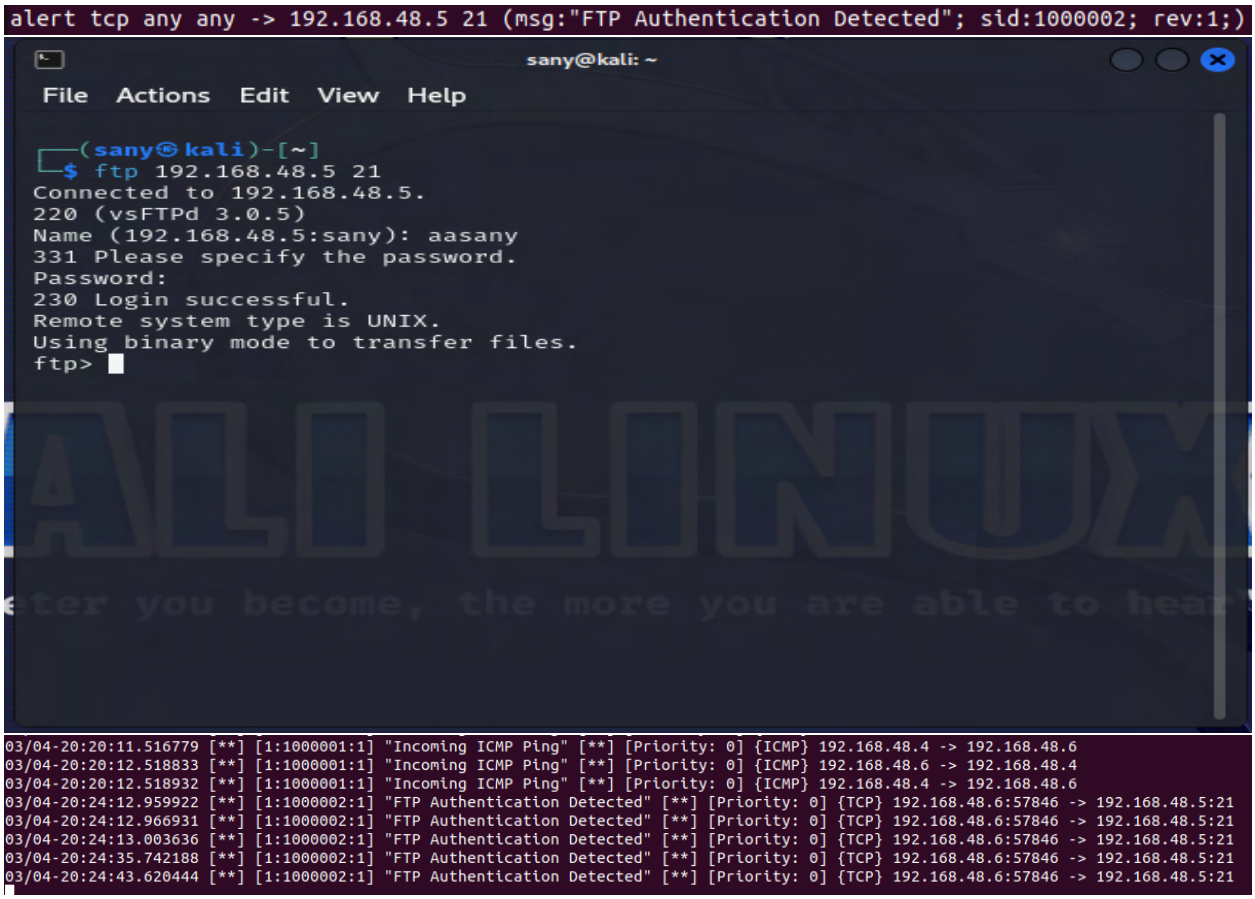
A ping command is executed from Kali Linux to test the connectivity to our Ubuntu 2.0

```
alert icmp any any -> $HOME_NET any (msg:"Incoming ICMP Ping"; sid:1000001; rev:1;)
```

```
sany@kali: ~
File  Actions  Edit  View  Help

┌──(sany㉿kali)-[~]
└─$ ping 192.168.48.4
PING 192.168.48.4 (192.168.48.4) 56(84) bytes of data.
64 bytes from 192.168.48.4: icmp_seq=1 ttl=64 time=8.76 ms
64 bytes from 192.168.48.4: icmp_seq=2 ttl=64 time=1.26 ms
64 bytes from 192.168.48.4: icmp_seq=3 ttl=64 time=1.31 ms
64 bytes from 192.168.48.4: icmp_seq=4 ttl=64 time=1.63 ms
64 bytes from 192.168.48.4: icmp_seq=5 ttl=64 time=1.41 ms
64 bytes from 192.168.48.4: icmp_seq=6 ttl=64 time=1.74 ms
64 bytes from 192.168.48.4: icmp_seq=7 ttl=64 time=2.06 ms
64 bytes from 192.168.48.4: icmp_seq=8 ttl=64 time=1.45 ms
64 bytes from 192.168.48.4: icmp_seq=9 ttl=64 time=1.06 ms
64 bytes from 192.168.48.4: icmp_seq=10 ttl=64 time=2.16 ms
64 bytes from 192.168.48.4: icmp_seq=11 ttl=64 time=1.48 ms
64 bytes from 192.168.48.4: icmp_seq=12 ttl=64 time=2.62 ms
64 bytes from 192.168.48.4: icmp_seq=13 ttl=64 time=1.64 ms
64 bytes from 192.168.48.4: icmp_seq=14 ttl=64 time=2.44 ms
64 bytes from 192.168.48.4: icmp_seq=15 ttl=64 time=1.49 ms
64 bytes from 192.168.48.4: icmp_seq=16 ttl=64 time=1.73 ms
64 bytes from 192.168.48.4: icmp_seq=17 ttl=64 time=1.51 ms
64 bytes from 192.168.48.4: icmp_seq=18 ttl=64 time=2.18 ms
64 bytes from 192.168.48.4: icmp_seq=19 ttl=64 time=1.77 ms
64 bytes from 192.168.48.4: icmp_seq=20 ttl=64 time=0.988 ms
64 bytes from 192.168.48.4: icmp_seq=21 ttl=64 time=1.41 ms
64 bytes from 192.168.48.4: icmp_seq=22 ttl=64 time=2.02 ms
^C
```

```
        transition memory: 1.73279
        fast pattern only: 7103
appid: MaxRss diff: 3072
appid: patterns loaded: 300
--------------------------------------------------
pcap DAQ configured to passive.
Commencing packet processing
++ [0] enp0s3
03/04-20:19:51.479088 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:51.480314 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:52.473968 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:52.474064 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:53.476162 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:53.476305 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:54.476426 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:54.476728 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:55.480828 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:55.480941 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:56.487052 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:56.487182 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:57.492289 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:57.492332 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:58.496871 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:58.497233 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:19:59.500029 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:19:59.500155 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:00.502021 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:00.502062 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:01.503009 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:01.503040 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:02.504923 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:02.505328 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:03.506273 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:03.506642 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:04.507230 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:04.507619 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:05.509126 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:05.509253 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:06.510967 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:06.511117 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:07.511331 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:07.511457 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:08.513816 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:08.513914 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:09.514517 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:09.514593 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
03/04-20:20:10.514979 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.6 -> 192.168.48.4
03/04-20:20:10.515019 [**] [1:1000001:1] "Incoming ICMP Ping" [**] [Priority: 0] {ICMP} 192.168.48.4 -> 192.168.48.6
```

To initiate an FTP (File Transfer Protocol) connection to Ubuntu 3.0 on port 21 from Kali Linux.



To establish an SSH connection to Metasploit2 with specific configuration options from Kali Linux.

```
alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Detected"; sid:1000003; rev:1;)
```

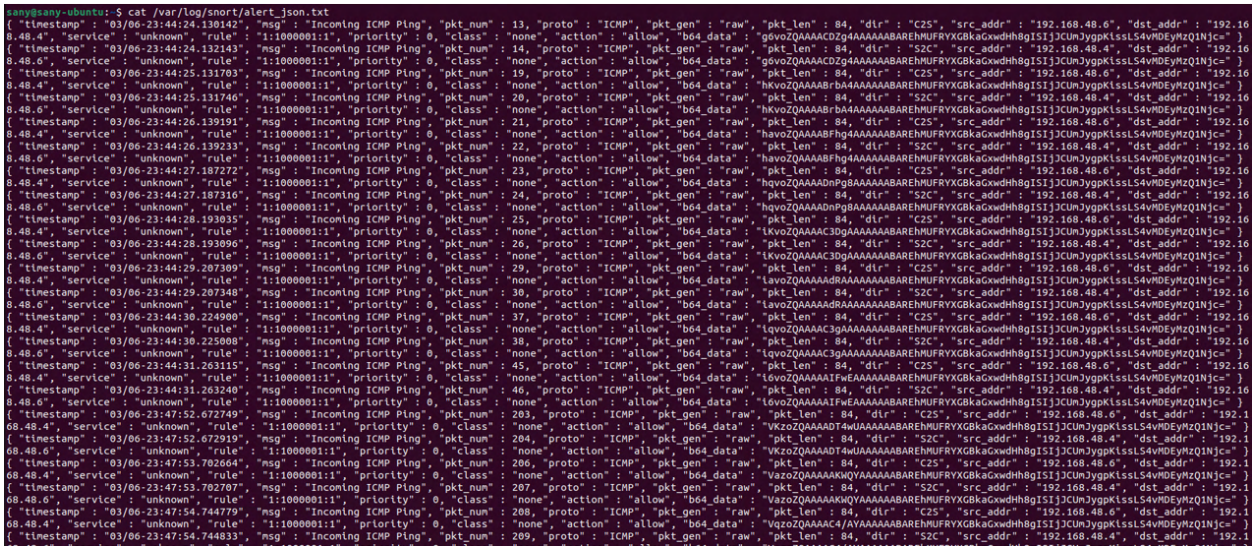
```
sany@kali: ~
File  Actions  Edit  View  Help

┌──(sany㉿kali)-[~]
└─$ ssh -oHostKeyAlgorithms=+ssh-dss msfadmin@192.168.48.7
msfadmin@192.168.48.7's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i68
6

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: Mon Mar  4 09:28:15 2024
msfadmin@metasploitable:~$ ls
vulnerable
msfadmin@metasploitable:~$
```

```
03/04-20:29:15.166788 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.166792 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.169470 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.174322 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.177469 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.182071 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.196917 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.212903 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.251078 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.253183 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:15.354845 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:24.251398 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:33834 -> 192.168.48.7:22
03/04-20:29:40.400142 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.403962 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.405229 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.433948 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.434052 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.436770 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.446647 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.462262 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
03/04-20:29:40.503059 [**] [1:1000003:1] "SSH Authentication Detected" [**] [Priority: 0] {TCP} 192.168.48.6:42654 -> 192.168.48.7:22
```

Snort3 IDS 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. Snort3 gives us the ability to log the intrusions in various file structure as we wish. For Snort to be an effective intrusion detection tool, it should log all alerts and store them on a local file or on a remote log server. Snort3 provides multiple options to log the Snort alerts. This latest update on Snort significantly improves the logging format that is compatible with the modern log management tools.

Logging in JSON structure



Logging in CSV structure

```
sany@sany-ubuntu:~$ cat /var/log/snort/alert_csv.txt
03/06-23:44:24.130142, 13, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:24.132143, 14, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:44:25.131703, 19, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:25.131746, 20, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:44:26.139191, 21, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:26.139233, 22, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:44:27.187272, 23, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:27.187316, 24, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:44:28.193035, 25, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:28.193096, 26, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:44:29.207309, 29, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:29.207348, 30, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:44:30.224900, 37, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:30.225008, 38, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:44:31.263115, 45, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:44:31.263240, 46, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:52.672749, 203, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:52.672919, 204, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:53.702664, 206, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:53.702707, 207, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:54.744779, 208, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:54.744833, 209, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:55.747360, 215, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:55.747415, 216, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:56.757205, 217, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:56.757259, 218, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:57.763793, 221, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:57.763845, 222, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:58.771767, 227, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:58.771840, 228, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:47:59.774567, 230, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:47:59.774634, 231, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:48:00.792593, 234, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:48:00.792670, 235, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:49:01.004605, 289, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:49:01.004654, 290, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:49:02.012332, 291, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:49:02.012395, 292, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:49:03.032328, 296, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:49:03.032383, 297, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:49:04.046414, 298, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:49:04.046464, 299, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:49:05.052345, 300, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:49:05.052451, 301, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:49:06.114669, 302, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:49:06.114714, 303, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
03/06-23:50:24.959439, 328, ICMP, raw, 84, C2S, 192.168.48.6:0, 192.168.48.4:0, 1:1000001:1, allow
03/06-23:50:24.959506, 329, ICMP, raw, 84, S2C, 192.168.48.4:0, 192.168.48.6:0, 1:1000001:1, allow
```

A problem about where to set up snort:

Problem Statement

If a subscriber configures Snort to operate as a sniffer, it will scan network packets and identify them. Snort can also log those packets to a disk file.To use Snort as a packet sniffer, users set the host's network interface to promiscuous mode to monitor all network traffic on the local network interface. But what happens when the packets are coming from outside the LAN?

Solution

When packets are coming from outside the LAN (Local Area Network), such as from the internet or another external network, Snort can still capture and analyze them if it is deployed in a position where it can see the traffic.

Here's what happens:

Placement: Snort needs to be placed in a network segment where it can see the traffic. This could be at a network perimeter, where the LAN connects to the internet, or within a demilitarized zone (DMZ) if the network architecture includes one.

Network Tap or Port Mirroring: Snort can capture packets coming from outside the LAN by using network taps or by configuring port mirroring on network switches. Network taps directly copy the traffic passing through a network segment to a monitoring interface where Snort can capture it. Port mirroring, also known as SPAN (Switched Port Analyzer) or RSPAN (Remote SPAN), repli-

cates traffic from one or more ports to another port where Snort is connected.

Promiscuous Mode: Snort's interface must still be set to promiscuous mode, regardless of whether the traffic is coming from inside or outside the LAN. This mode allows the network interface to capture all packets on the network segment, not just those intended for the host running Snort.

Analysis: Once Snort captures the packets, it can analyze them using its rulesets to detect any suspicious or malicious activity. This includes traffic originating from outside the LAN that may be attempting to exploit vulnerabilities or perform unauthorized activities.

In summary, while Snort is typically deployed within LANs to monitor internal traffic, it can also be configured to monitor and analyze traffic coming from outside the LAN by being placed strategically within the network architecture and using appropriate capture methods like network taps or port mirroring.

5 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.