🛡️ IDS/IPS Lab Report Using Snort on Parrot OS

A. Setting Up the Lab Environment

I launched UTM on my MacBook and booted into my Parrot OS virtual machine. Once the system was up, I opened the terminal and updated all packages using the following command:

bash

CopyEdit

```
sudo apt update && sudo apt upgrade -y
```

📸 *Screenshot: Terminal showing system update and upgrade completed.*

B. Installing and Configuring the IDS/IPS

1. Installing Snort

I installed Snort using the default package manager in Parrot OS by running:

bash

CopyEdit

```
sudo apt install snort -y
```

Snort installed successfully without any errors.

2. Adding Custom Detection Rules

After installation, I navigated to the Snort rules directory and opened the local.rules file:

bash

CopyEdit

```
sudo nano /etc/snort/rules/local.rules
```

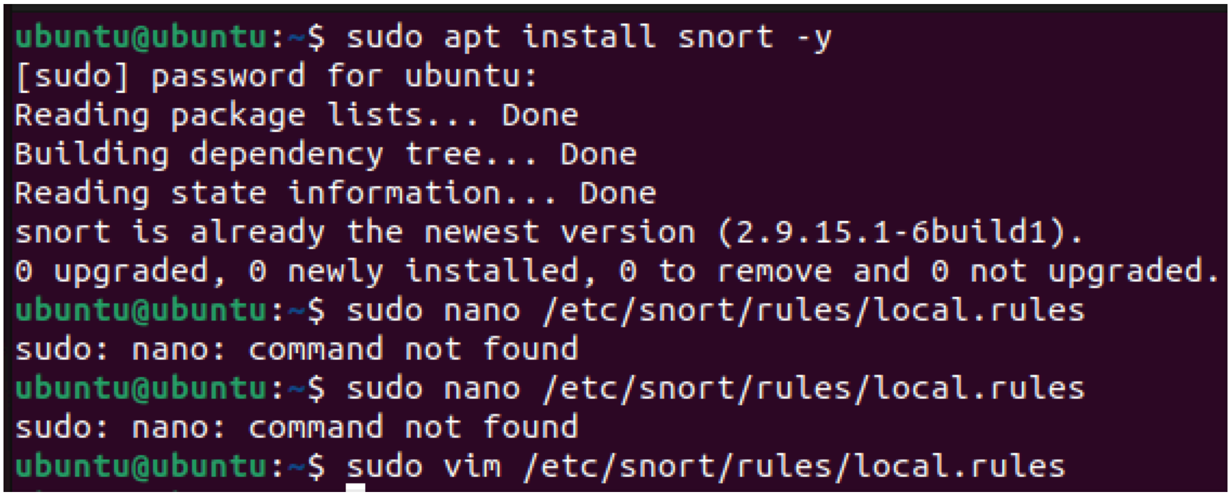
Inside the file, I added two custom rules:

python

CopyEdit

```
alert icmp any any -> any any (msg:"Custom Alert: Ping Detected"; sid:1000001;)
alert tcp any any -> any 23 (msg:"Custom Alert: Telnet Detected"; sid:1000002;)
```

I saved and exited the file.



```
ubuntu@ubuntu:~$ sudo apt install snort -y
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snort is already the newest version (2.9.15.1-6build1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ubuntu:~$ sudo nano /etc/snort/rules/local.rules
sudo: nano: command not found
ubuntu@ubuntu:~$ sudo nano /etc/snort/rules/local.rules
sudo: nano: command not found
ubuntu@ubuntu:~$ sudo vim /etc/snort/rules/local.rules
```

C. Testing the Rules

1. Running Snort

I launched Snort in console alert mode using my interface enp0s1:

bash

CopyEdit

```
sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s1
```

Snort started successfully, monitoring traffic on interface enp0s1.

2. Triggering the Alerts

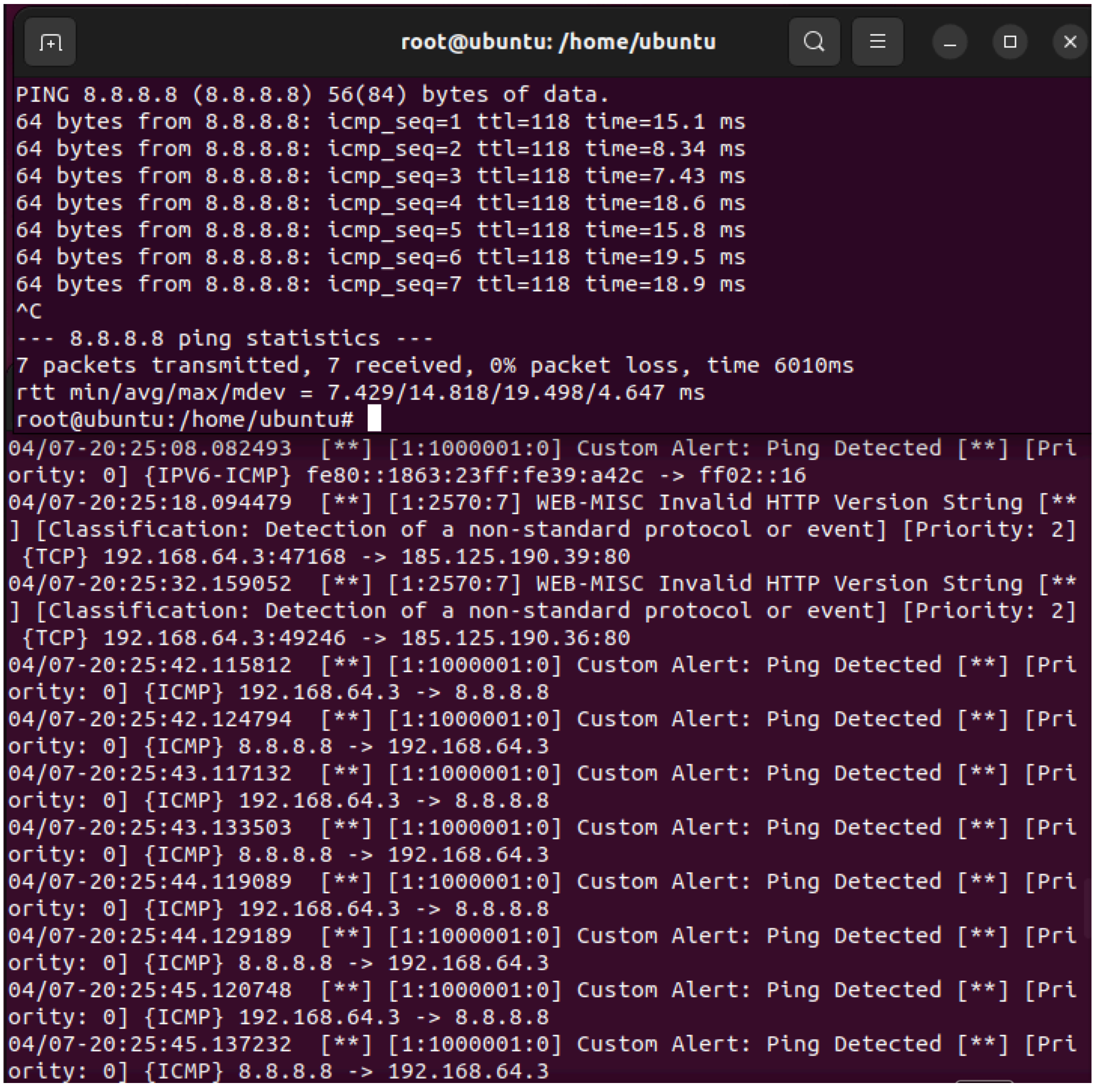
In a separate terminal window, I triggered both rules:

- For ICMP (ping) detection:
  bash
  CopyEdit

  ```
  ping 8.8.8.8
  ```
-
- For Telnet traffic detection:
  bash
  CopyEdit

  ```
  telnet localhost 23
  ```
-

Both activities triggered alerts in the Snort console as expected.



```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=15.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=8.34 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=7.43 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=118 time=18.6 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=118 time=15.8 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=118 time=19.5 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=118 time=18.9 ms
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6010ms
rtt min/avg/max/mdev = 7.429/14.818/19.498/4.647 ms
root@ubuntu:/home/ubuntu#
```

```
04/07-20:25:08.082493  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {IPV6-ICMP} fe80::1863:23ff:fe39:a42c -> ff02::16
04/07-20:25:18.094479  [**] [1:2570:7] WEB-MISC Invalid HTTP Version String [**
] [Classification: Detection of a non-standard protocol or event] [Priority: 2]
 {TCP} 192.168.64.3:47168 -> 185.125.190.39:80
04/07-20:25:32.159052  [**] [1:2570:7] WEB-MISC Invalid HTTP Version String [**
] [Classification: Detection of a non-standard protocol or event] [Priority: 2]
 {TCP} 192.168.64.3:49246 -> 185.125.190.36:80
04/07-20:25:42.115812  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 192.168.64.3 -> 8.8.8.8
04/07-20:25:42.124794  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 8.8.8.8 -> 192.168.64.3
04/07-20:25:43.117132  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 192.168.64.3 -> 8.8.8.8
04/07-20:25:43.133503  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 8.8.8.8 -> 192.168.64.3
04/07-20:25:44.119089  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 192.168.64.3 -> 8.8.8.8
04/07-20:25:44.129189  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 8.8.8.8 -> 192.168.64.3
04/07-20:25:45.120748  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 192.168.64.3 -> 8.8.8.8
04/07-20:25:45.137232  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 8.8.8.8 -> 192.168.64.3
```

D. Rule Tuning: Before & After Comparison

1. Before Tuning

Initially, when I ran the ping command several times, Snort generated an alert for every packet. This caused the console to fill up with repeated alerts.

2. After Tuning the ICMP Rule

To reduce alert spam, I added a threshold to the ICMP rule. I reopened the rules file:

bash

CopyEdit

```
sudo nano /etc/snort/rules/local.rules
```

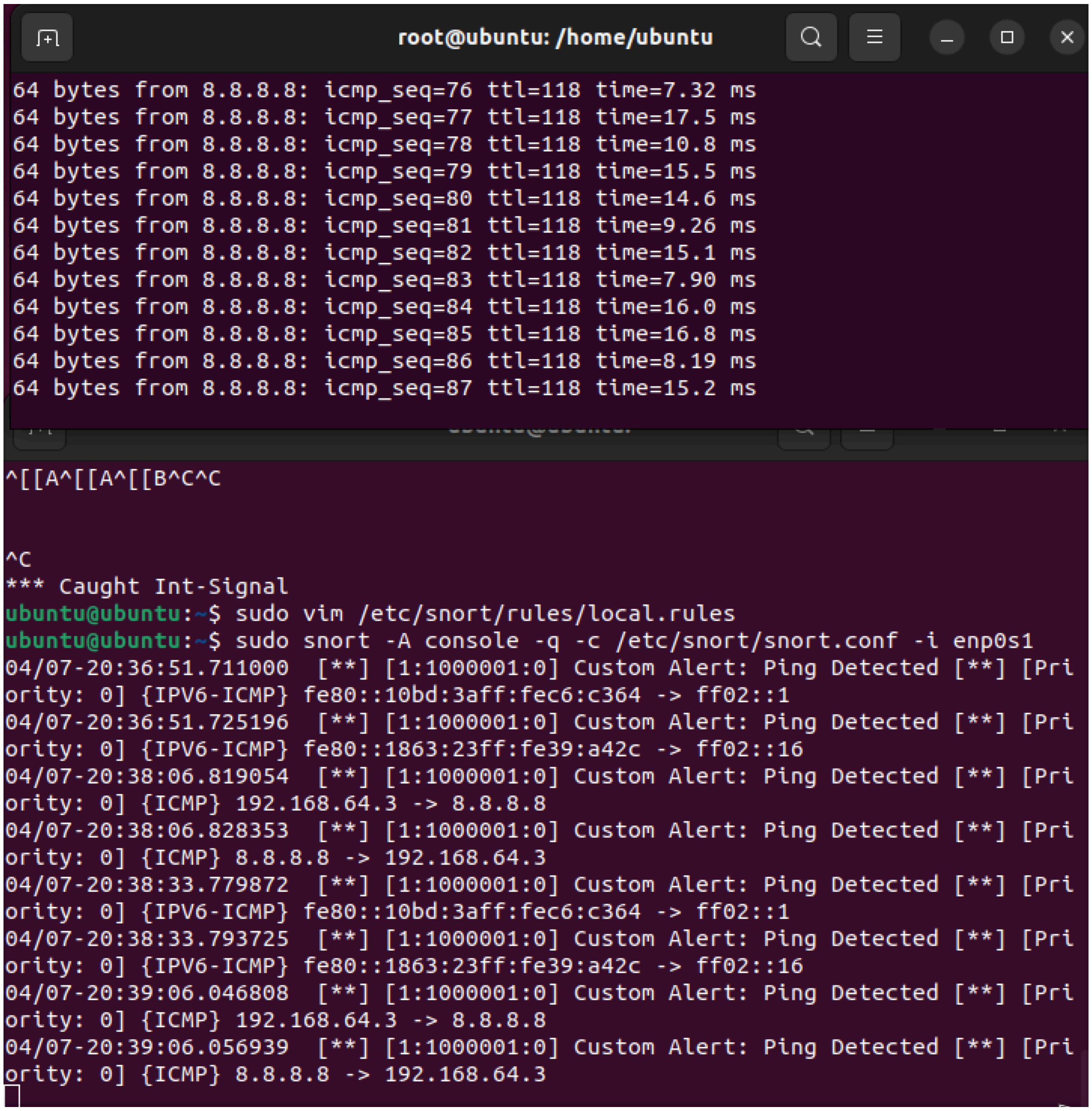
I modified the ping detection rule to:

pgsql

CopyEdit

```
alert icmp any any -> any any (msg:"Custom Alert: Ping Detected"; sid:1000001;
threshold: type limit, track by_src, count 1, seconds 60;)
```

After saving the changes, I restarted Snort and ran the same ping test. This time, only one alert appeared per minute for each source IP, significantly reducing the noise.



```
64 bytes from 8.8.8.8: icmp_seq=76 ttl=118 time=7.32 ms
64 bytes from 8.8.8.8: icmp_seq=77 ttl=118 time=17.5 ms
64 bytes from 8.8.8.8: icmp_seq=78 ttl=118 time=10.8 ms
64 bytes from 8.8.8.8: icmp_seq=79 ttl=118 time=15.5 ms
64 bytes from 8.8.8.8: icmp_seq=80 ttl=118 time=14.6 ms
64 bytes from 8.8.8.8: icmp_seq=81 ttl=118 time=9.26 ms
64 bytes from 8.8.8.8: icmp_seq=82 ttl=118 time=15.1 ms
64 bytes from 8.8.8.8: icmp_seq=83 ttl=118 time=7.90 ms
64 bytes from 8.8.8.8: icmp_seq=84 ttl=118 time=16.0 ms
64 bytes from 8.8.8.8: icmp_seq=85 ttl=118 time=16.8 ms
64 bytes from 8.8.8.8: icmp_seq=86 ttl=118 time=8.19 ms
64 bytes from 8.8.8.8: icmp_seq=87 ttl=118 time=15.2 ms
```

```
^[[A^[[A^[[B^C^C


^C
*** Caught Int-Signal
ubuntu@ubuntu:~$ sudo vim /etc/snort/rules/local.rules
ubuntu@ubuntu:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s1
04/07-20:36:51.711000  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {IPV6-ICMP} fe80::10bd:3aff:fec6:c364 -> ff02::1
04/07-20:36:51.725196  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {IPV6-ICMP} fe80::1863:23ff:fe39:a42c -> ff02::16
04/07-20:38:06.819054  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 192.168.64.3 -> 8.8.8.8
04/07-20:38:06.828353  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 8.8.8.8 -> 192.168.64.3
04/07-20:38:33.779872  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {IPV6-ICMP} fe80::10bd:3aff:fec6:c364 -> ff02::1
04/07-20:38:33.793725  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {IPV6-ICMP} fe80::1863:23ff:fe39:a42c -> ff02::16
04/07-20:39:06.046808  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 192.168.64.3 -> 8.8.8.8
04/07-20:39:06.056939  [**] [1:1000001:0] Custom Alert: Ping Detected [**] [Pri
ority: 0] {ICMP} 8.8.8.8 -> 192.168.64.3
```

E. Defense Automation Script

1. Creating the Script

I created a script to block IP addresses using iptables. I created a new file:

bash

CopyEdit

```
nano block_ip.sh
```

I added the following content:

bash

CopyEdit

```
#!/bin/bash
# This script blocks an IP address using iptables

if [ -z "$1" ]; then
   echo "Usage: $0 <IP_Address>"
```

```
    exit 1
fi

sudo iptables -A INPUT -s $1 -j DROP
echo "IP $1 has been blocked."
```

2. Making It Executable

To make the script runnable, I used:

bash

CopyEdit

```
chmod +x block_ip.sh
```

3. Testing the Script

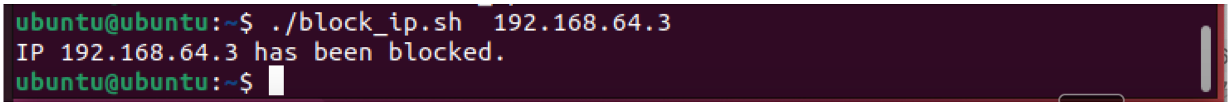
I tested the script by blocking a sample IP:

bash

CopyEdit

```
./block_ip.sh 192.168.64.5
```

The script successfully blocked the IP and printed a confirmation message.



```
ubuntu@ubuntu:~$ ./block_ip.sh  192.168.64.3
IP 192.168.64.3 has been blocked.
ubuntu@ubuntu:~$ 
```