BUILDING A SECURE NETWORK WITH IDS/IPS

Date: June 2025
Prepared By: Sahithi Gudigopuram

Table of Contents

S.No	Title	Page Numbers
1.	Project Overview	03
1.1	Project title	03
1.2	Objective / Purpose	03
1.3	Scope	03
2.	Tools & Technologies	03
3.	Network Topology	04
4.	System Configuration and System Setup	06
4.1	UFW Firewall Setup on Ubuntu	07
4.2	Snort Installation and Verification	07
4.3	Static IP Configuration for Ubuntu	07
4.4	Static IP Configuration for Kali	09
5.	Attack Simulations & IDS/IPS Response	10
5.1	TCP SYN Scan Detection	10
5.2	SSH Brute Force Attack	11
5.3	SYN Flood Attack (DDoS)	14
5.4	Spoofed ICMP Echo Request	15
6.	Traffic Capture & Log Analysis	16
6.1	Network capturer and analysis	16
6.2	SYN Flood Log Analysis Using Snort and Wireshark	17
7.	fail2ban Configuration & Jail Testing	20
8.	Adding Custom Rules to Reduce False Positives	22
9.	Key Observations	22
10.	Conclusion	23

1. Project Overview

1.1. Project Title: Implementation of a Network Intrusion Detection and Prevention Lab Using Snort and Fail2ban

1.2. Objective / Purpose: The primary objective of this project is to design, configure, and test a virtual lab environment capable of detecting and responding to common network-based attacks in real time. The goal is to deploy Snort as an Intrusion Detection System (IDS) and fail2ban as a lightweight Intrusion Prevention System (IPS), validating both detection accuracy and automated mitigation through live simulations.

1.3.Scope: This project covers the setup and configuration of a secure network topology using:

- Static IP addressing for stability
- Snort for traffic analysis and threat detection
- Fail2ban for dynamic response and IP banning
- Kali Linux to simulate real-world attack vectors (e.g., ping sweeps, Nmap scans, SSH brute-force)
- Rule customisation and tuning to reduce false positives.

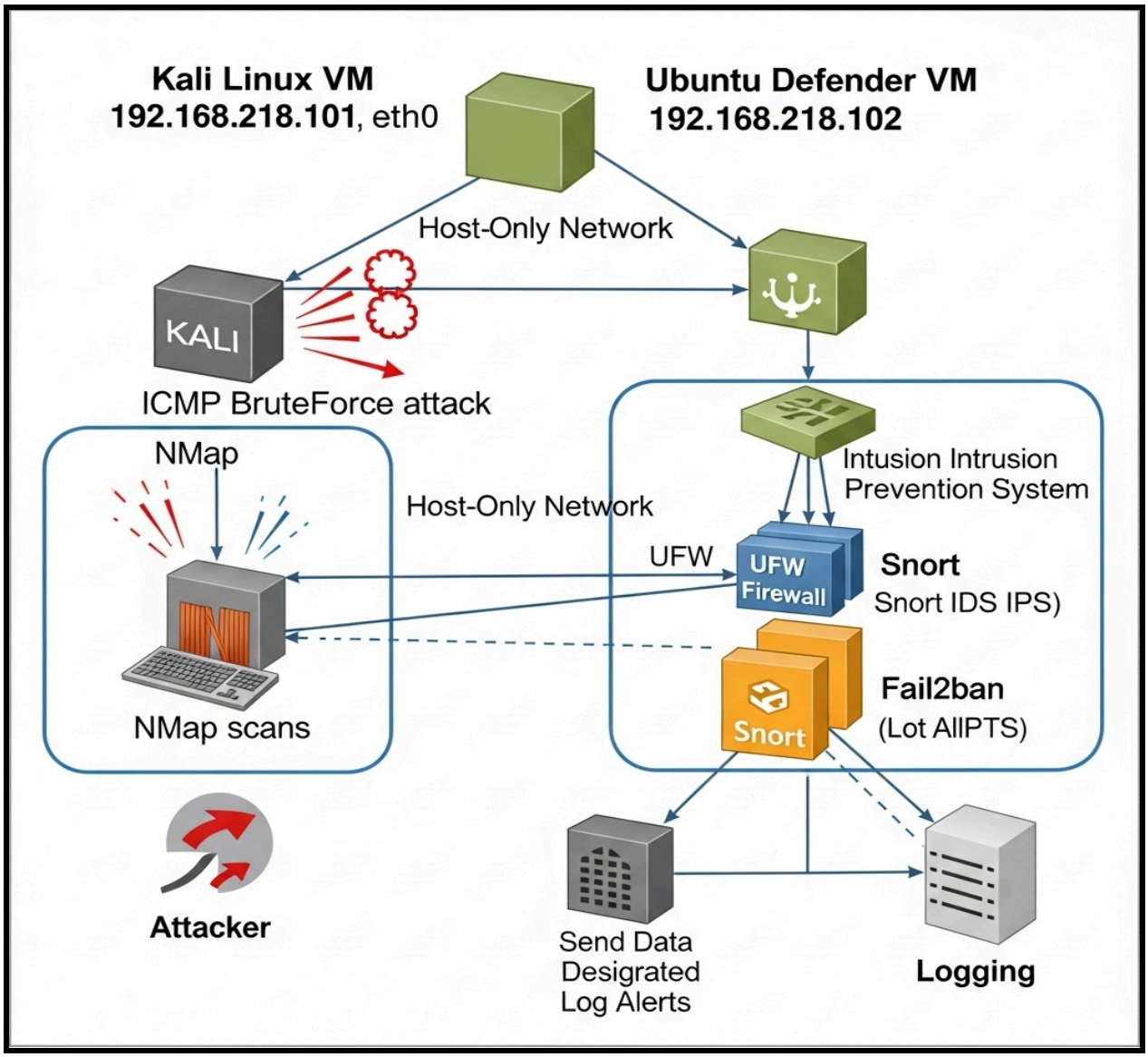
2. Tools & Technologies Used

- **Operating Systems:** Ubuntu (IDS/IPS host), Kali Linux (attacker simulation)
- **Intrusion Detection System (IDS):** Snort v2.x
- **Intrusion Prevention / Response:** fail2ban v1.x
- **UFW** (Uncomplicated Firewall) for traffic filtering and access control
- **Offensive Security Tools (Testing & Validation):**nmap (port scanning), hydra (brute-force testing)
- **Configuration Files Used:**
- /etc/snort/snort.conf
- /etc/snort/rules/local.rules
- /etc/fail2ban/jail.local

3. Network Topology

Description of system setup

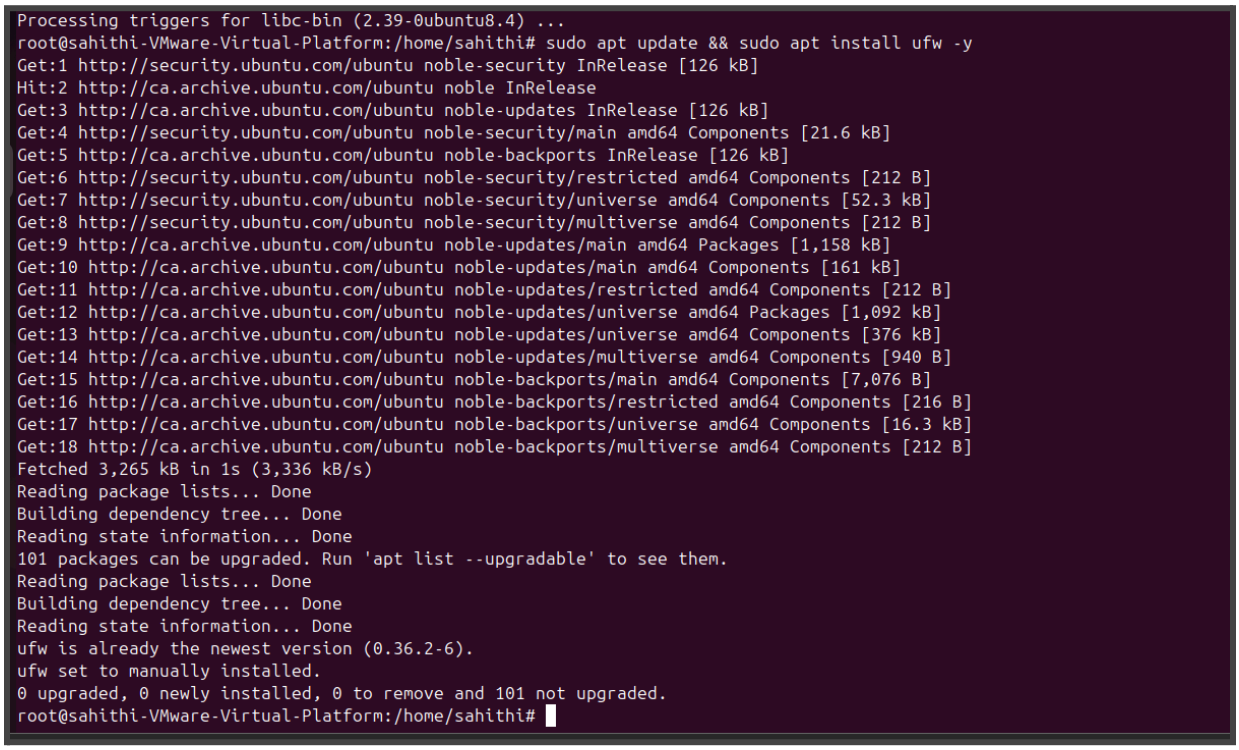
Host	Role	IP address	Interface
Ubuntu	IDS/IPS	192.168.218.102	ens33
Kali	Attacker	192.168.218.101	eth0


Kali Linux VM
192.168.218.101, eth0
Ubuntu Defender VM
192.168.218.102
Host-Only Network
KALI
ICMP BruteForce attack
NMap
Host-Only Network
Intusion Intrusion
Prevention System
UFW
UFW
Firewall
Snort
Snort IDS IPS)
Snort
Fail2ban
(Lot AllPTS)
NMap scans
Attacker
Send Data
Desigrated
Log Alerts
Logging

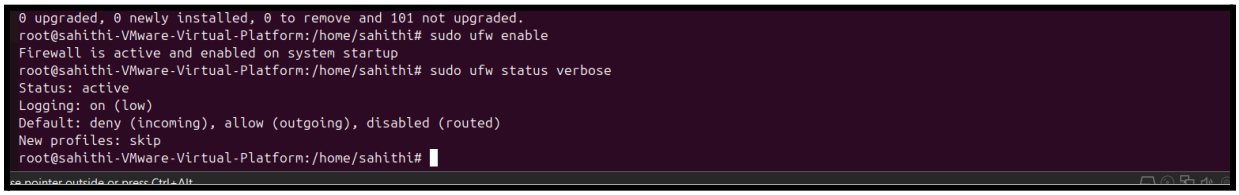
4. System Configuration and System Setup

4.1. UFW Firewall Setup on Ubuntu

- Step 1:Install and enable UFW: “ sudo apt update && sudo apt install ufw -y”

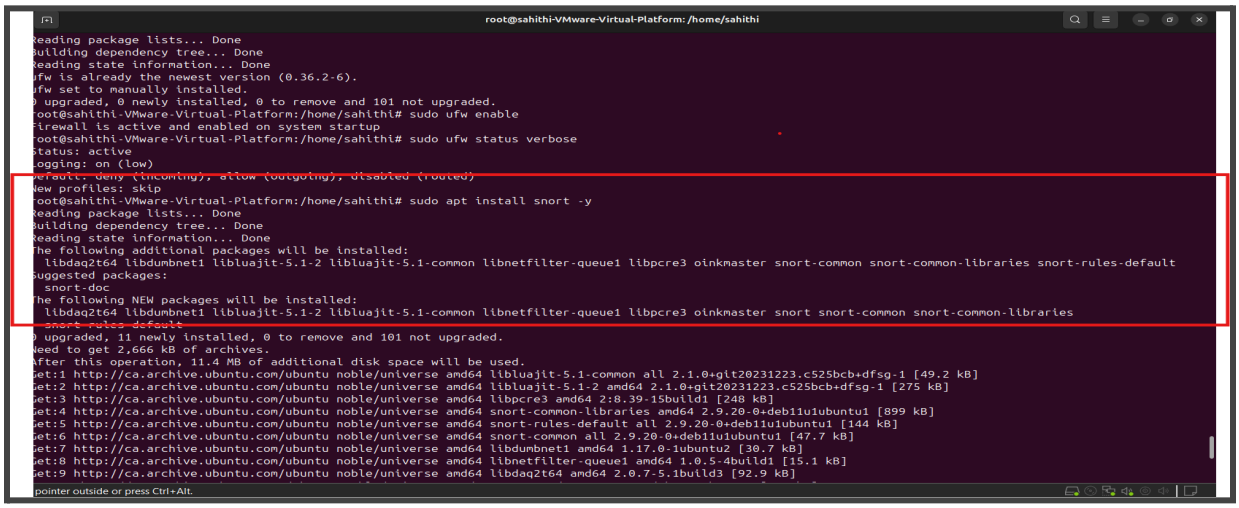


- Step 2: We need to enable and check the status “sudo ufw enable
  sudo ufw status verbose”



4.2 Snort Installation and Verification

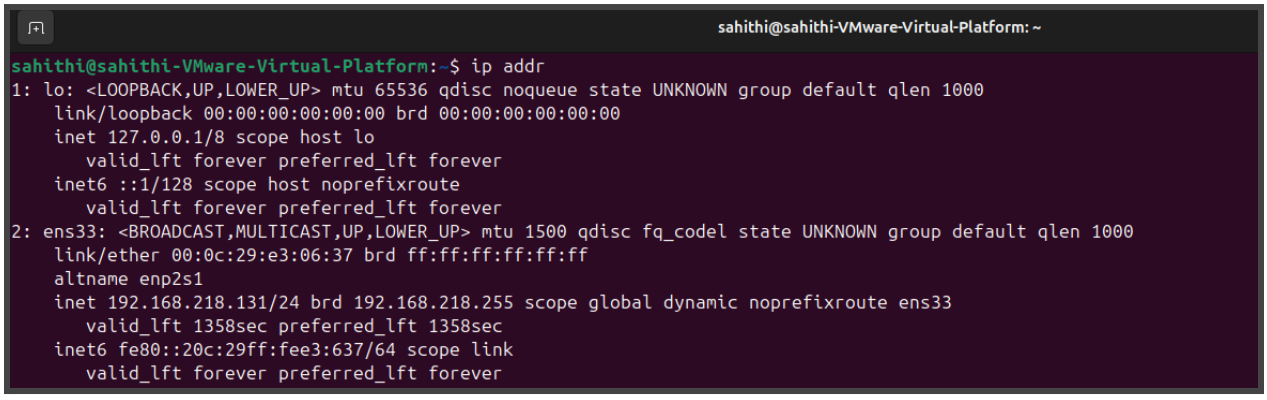
- Step 1: Install Snort “sudo apt install snort -y”



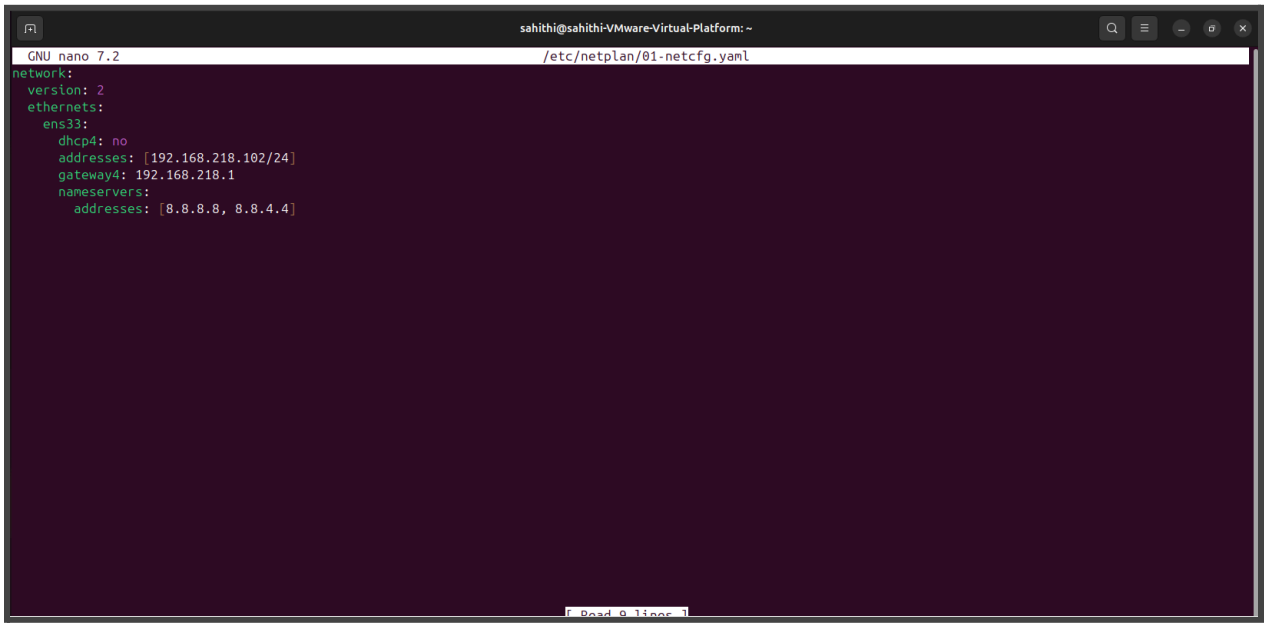
- Step 2: We need to test “ snort -V”

4.3. Static IP Configuration for Ubuntu

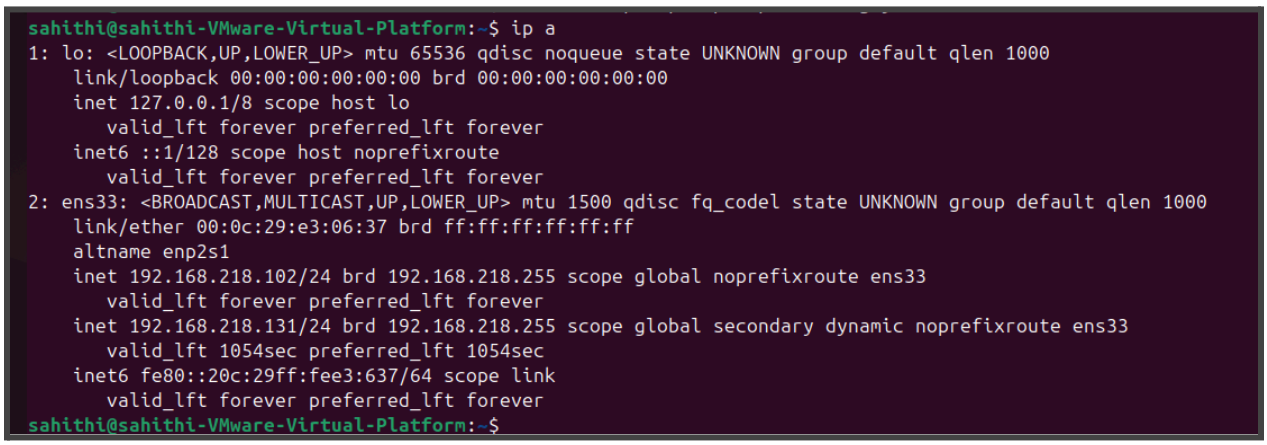
- Step 1: We need to set our VM network to Host-Only
- Step 2: Check Identity Interface “ Ip a” and My Ubuntu is using the ens33 network interface 192.168.218.131.



- Step 3: We need to configure Netplan” sudo nano /etc/netplan/01-netcfg.yaml”

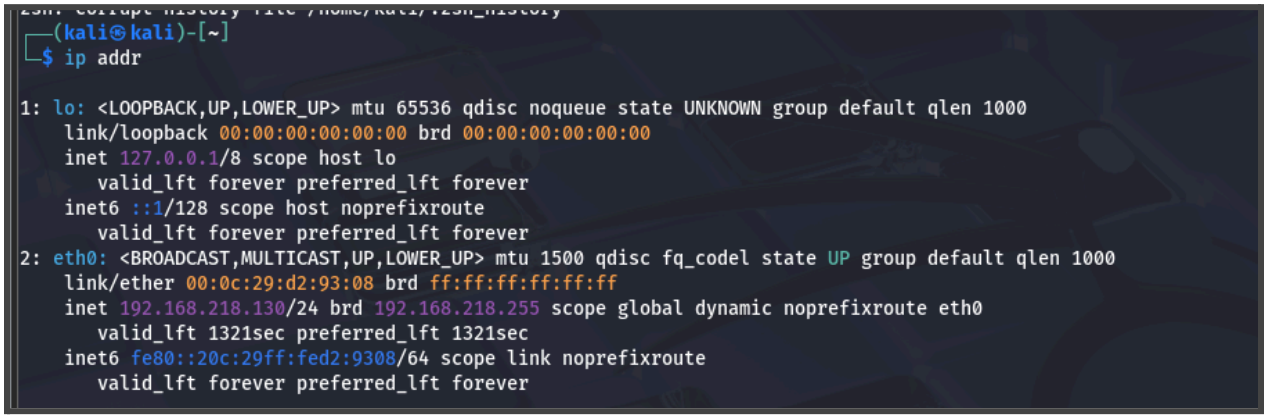


- Step 4: Apply Changes “ sudo netplan apply”
- Step 5: Now we need to check the “ip a” and my static ip is 192.168.218.102/24

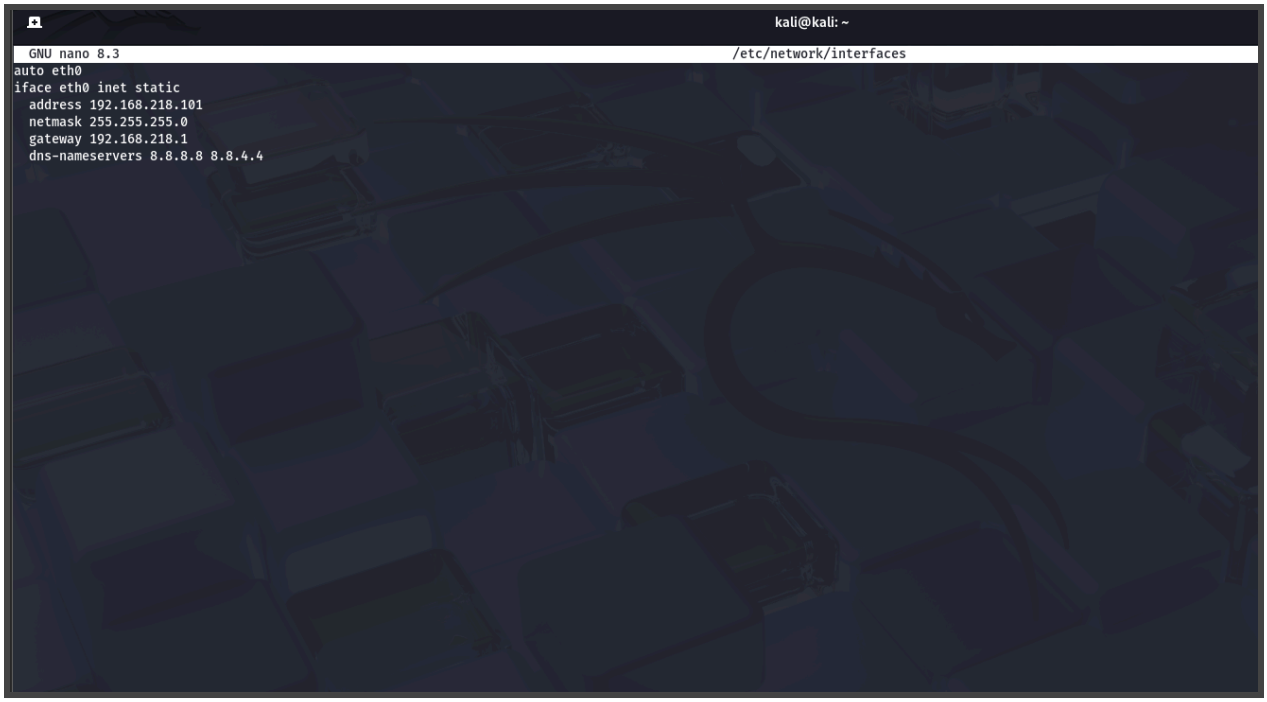


4.4. Static IP Configuration for Kali Linux

- Step 1: Check Identity Interface “ Ip a:



- Step 2: Edit the interfaces file “ sudo nano /etc/network/interfaces”



- Step 3: Restart the networking service “ sudo systemctl restart networking” and check the “ip a” and my static IP is 192.168.192.101.

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d2:93:08 brd ff:ff:ff:ff:ff:ff
    inet 192.168.218.130/24 brd 192.168.218.255 scope global dynamic noprefixroute eth0
       valid_lft 912sec preferred_lft 912sec
    inet 192.168.218.101/24 brd 192.168.218.255 scope global secondary eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed2:9308/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

5. Attack Simulations & IDS/IPS Response

5.1.TCP SYN Scan Detection:

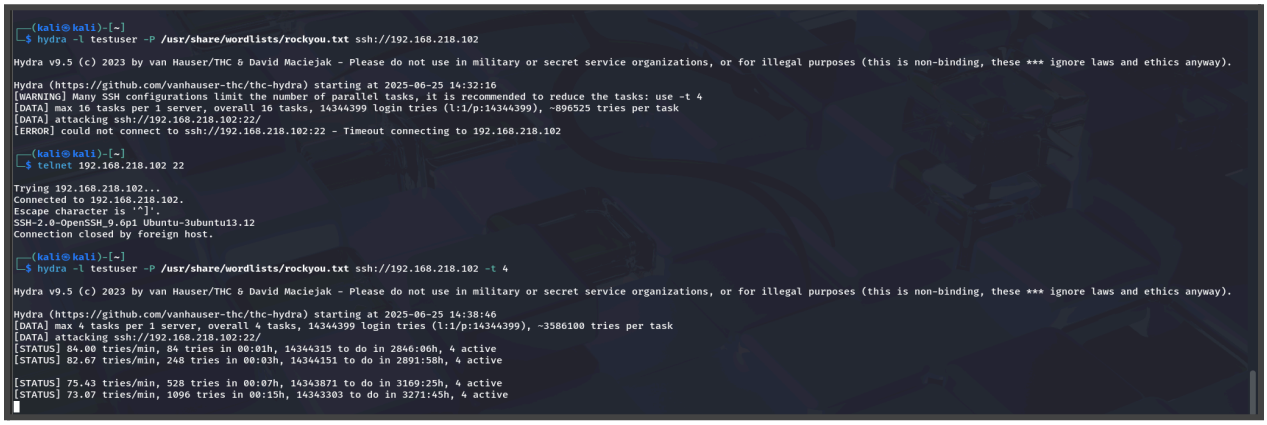
- A TCP SYN scan (nmap -sS) was executed from Kali to the target Ubuntu.
- This mimics stealthy reconnaissance techniques used by attackers to detect open ports.
- Snort was run in console alert mode on interface `ens33`.
- Used default rule sets defined in `/etc/snort/snort.conf`.
- Snort identified SNMP-related traffic on TCP ports 705 and 161.
- Alerts were labelled as “Attempted Information Leak”, suggesting unauthorised probing behaviour.
- UFW successfully filtered all scanned ports, providing passive protection while Snort logged the scanning attempt.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sS 192.168.218.102

Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-25 14:14 EDT
Nmap scan report for 192.168.218.102
Host is up (0.00057s latency).
All 1000 scanned ports on 192.168.218.102 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:0C:29:53:C6:4A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 34.32 seconds

┌──(kali㉿kali)-[~]
```

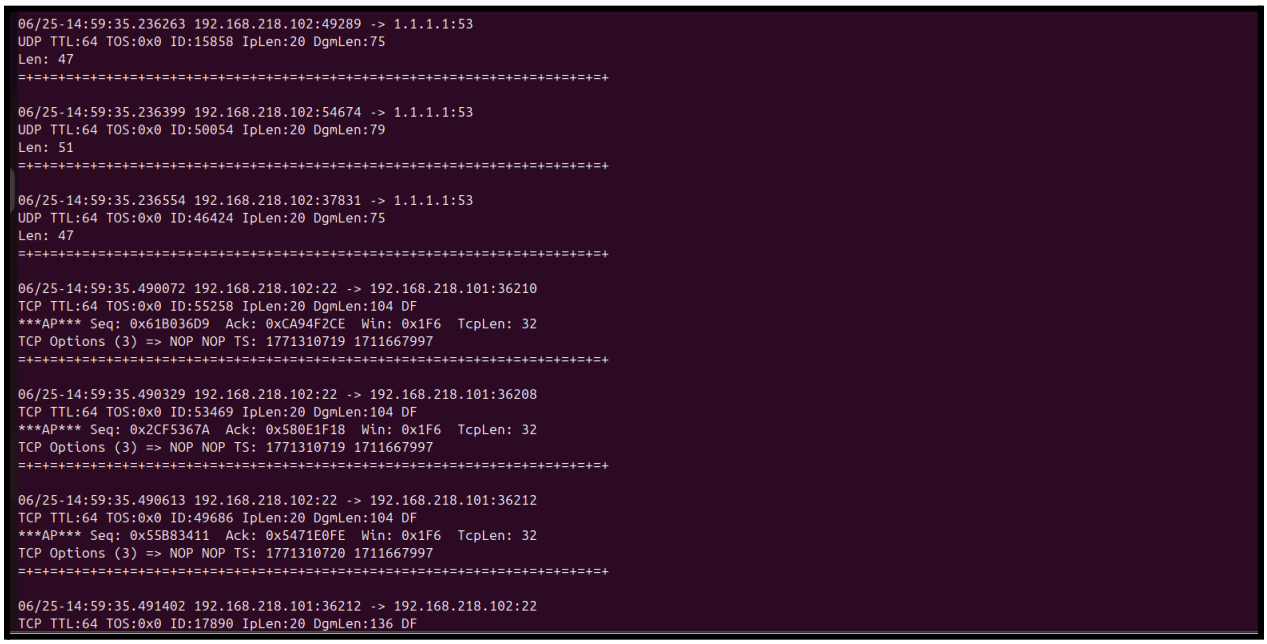
```
root@sahithi-VMware-Virtual-Platform:/home/sahithi# sudo snort -i ens33 -A console -q -c /etc/snort/snort.conf
06/25-14:14:20.129314  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.218.101:60676 -> 192.168.218.102:705
06/25-14:14:20.229839  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.218.101:60678 -> 192.168.218.102:705
06/25-14:14:24.579737  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.218.101:60676 -> 192.168.218.102:161
06/25-14:14:24.679908  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.218.101:60678 -> 192.168.218.102:161
```

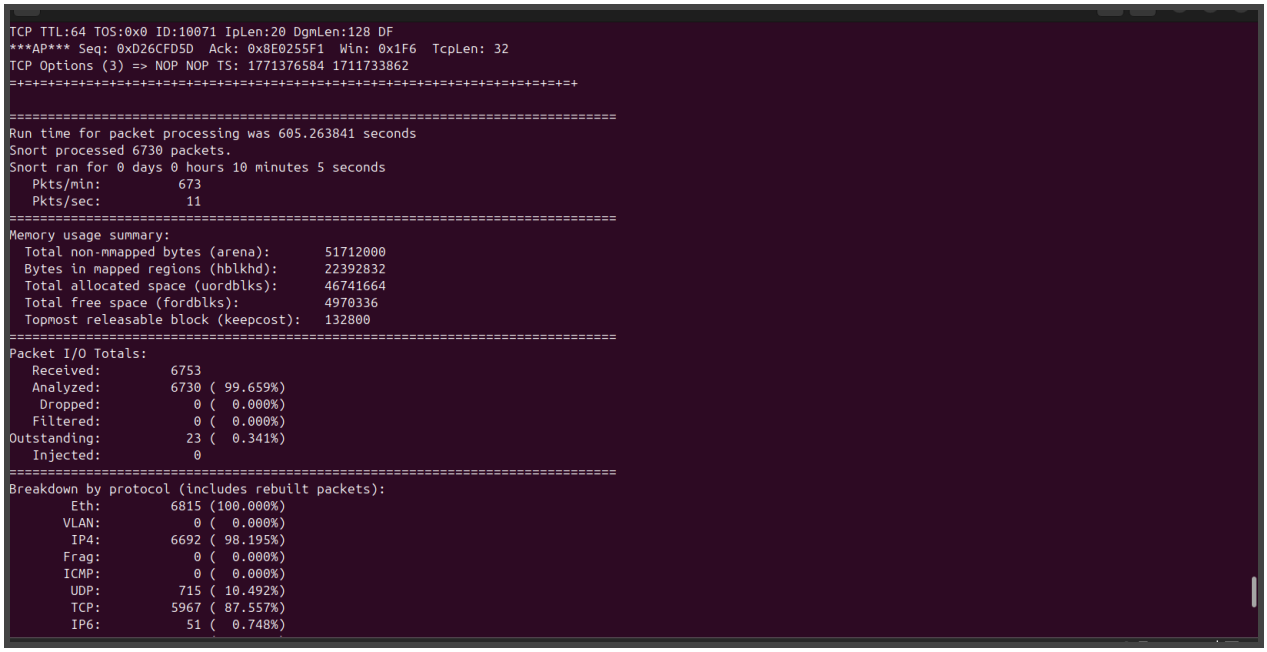
5.2. SSH Brute Force Attack

- I used Hydra for the brute force attract, so I installed it “sudo apt install hydra”
- We need to enable the wordlist “ sudo gzip -d /usr/share/wordlists/rockyou.txt.gz”
- Now we need to run the attack “
  hydra -l testuser -P /usr/share/wordlists/rockyou.txt ssh://192.168.218.102 -t 4”



- Start running the snort in verbose “ sudo snort -i ens33 -v -A console -c /etc/snort/snort.conf”
- This screenshot shows Snort capturing live SSH brute-force traffic between Kali (192.168.218.101) and Ubuntu IDS (192.168.218.102). The logs highlight TCP packets targeting port 22 with a series of connection attempts. Response packets (AP) from Ubuntu confirm active SSH handshake sequences.

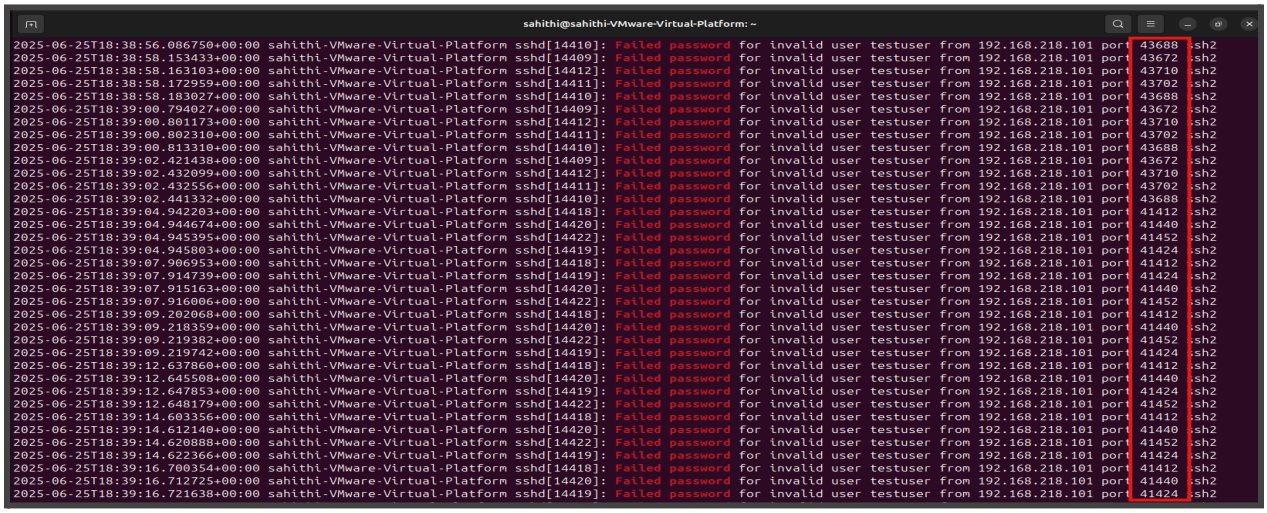




- To catch SSH login attempts, we need to set the rules “ sudo nano /etc/snort/rules/local.rules”
- Add this rule “alert tcp any any -> 192.168.218.102 22 (msg:"SSH Brute-Force Attempt Detected"; flags:S; threshold:type threshold, track by_src, count 1, seconds 5; sid:1000003; rev:1;)”
- Multiple alerts are generated in real time
- Each alert indicated 1or more connection attempts from the attacker's IP within a 5-second window.
- Traffic was verified with packet data showing consistent SYN activity.

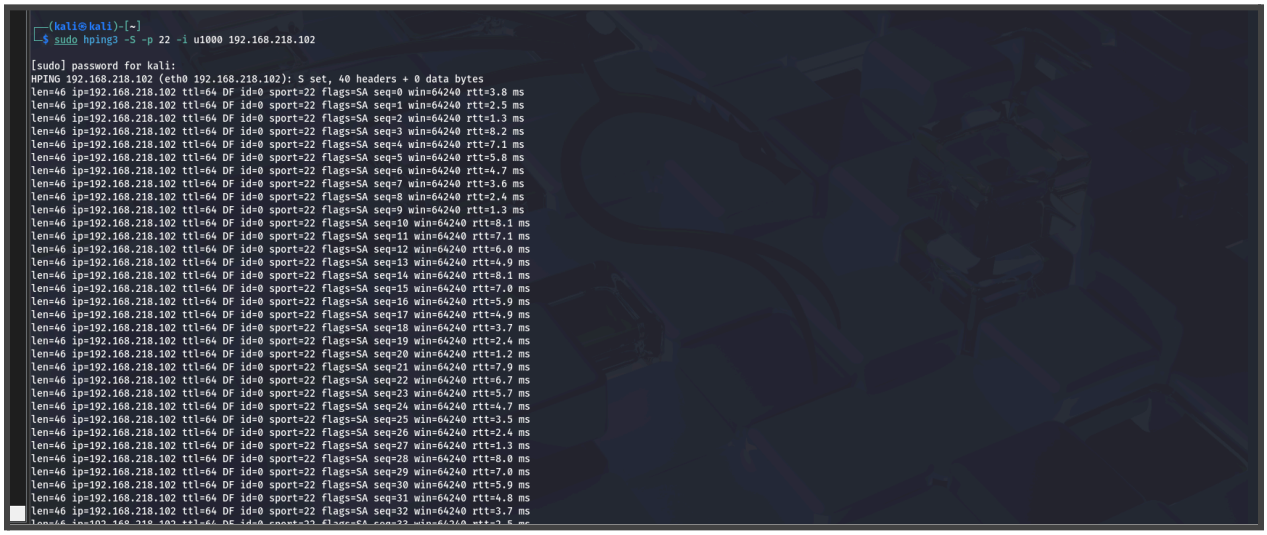
```
^C*** Caught Int-Signal
root@sahithi-VMware-Virtual-Platform:/home/sahithi# sudo nano /etc/snort/rules/local.rules
root@sahithi-VMware-Virtual-Platform:/home/sahithi# sudo snort -i ens33 -A console -q -c /etc/snort/snort.conf
06/25-15:26:00.632306  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:34650 -> 192.168.218.102:22
06/25-15:26:00.967565  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:34654 -> 192.168.218.102:22
06/25-15:26:00.968537  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:34660 -> 192.168.218.102:22
06/25-15:26:00.969719  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:34664 -> 192.168.218.102:22
06/25-15:26:00.969719  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:34680 -> 192.168.218.102:22
06/25-15:26:15.181052  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:42750 -> 192.168.218.102:22
06/25-15:26:15.191260  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:42766 -> 192.168.218.102:22
06/25-15:26:15.201905  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:42768 -> 192.168.218.102:22
06/25-15:26:15.206135  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:42772 -> 192.168.218.102:22
06/25-15:26:34.083265  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:38260 -> 192.168.218.102:22
06/25-15:26:34.107390  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:38262 -> 192.168.218.102:22
06/25-15:26:34.116152  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:38276 -> 192.168.218.102:22
06/25-15:26:34.128566  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:38292 -> 192.168.218.102:22
06/25-15:26:55.346765  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:46076 -> 192.168.218.102:22
06/25-15:26:57.719944  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:46078 -> 192.168.218.102:22
06/25-15:26:57.730537  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:46090 -> 192.168.218.102:22
06/25-15:26:57.740281  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:46106 -> 192.168.218.102:22
06/25-15:27:11.910213  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:54982 -> 192.168.218.102:22
06/25-15:27:16.628714  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:53392 -> 192.168.218.102:22
06/25-15:27:16.638989  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:53398 -> 192.168.218.102:22
06/25-15:27:16.649347  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:53412 -> 192.168.218.102:22
06/25-15:27:30.814921  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:54088 -> 192.168.218.102:22
06/25-15:27:35.564731  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:32888 -> 192.168.218.102:22
06/25-15:27:35.575670  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:32892 -> 192.168.218.102:22
06/25-15:27:35.586894  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:32896 -> 192.168.218.102:22
06/25-15:27:54.444572  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:39044 -> 192.168.218.102:22
06/25-15:27:59.182993  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:39048 -> 192.168.218.102:22
06/25-15:27:59.194630  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:39062 -> 192.168.218.102:22
06/25-15:27:59.204829  [**] [1:1000003:1] SSH Brute-Force Attempt Detected [**] [Priority: 0] {TCP} 192.168.218.101:39064 -> 192.168.218.102:22
```

- **Log analysis:** We are finding the password failed logins “grep "Failed password'/var/log/auth.log”

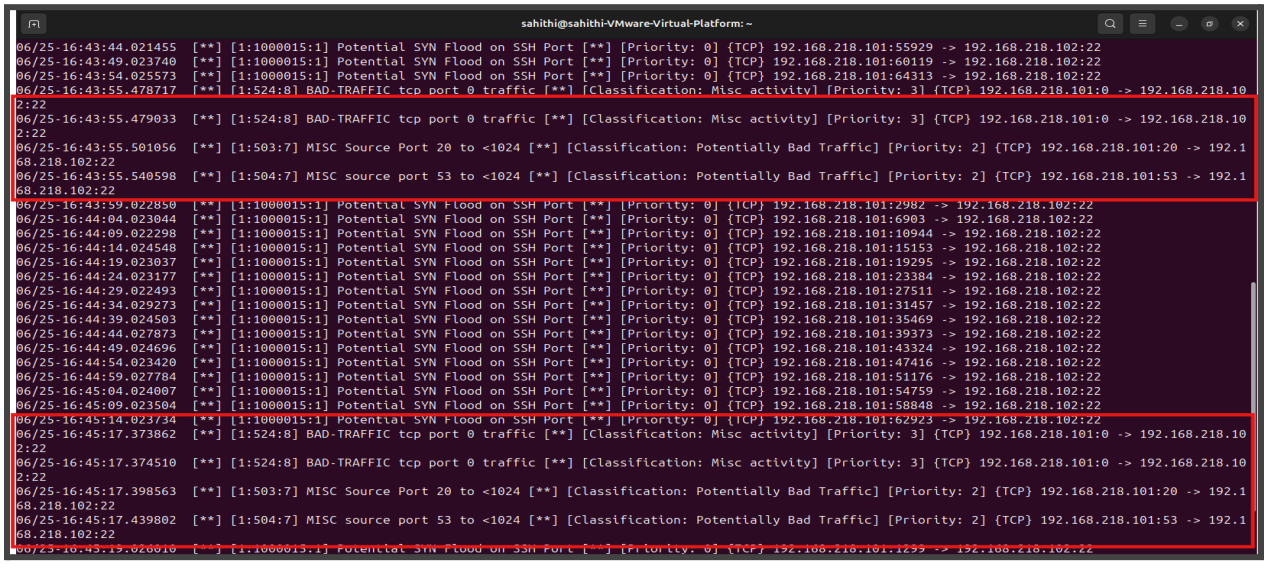


5.3 SYN Flood Attack (DDoS)

- First, we need to update the Snort rules “ sudo nano /etc/snort/rules/local.rules”
- I am adding a new rule to the IDS “ alert tcp any any -> 192.168.218.102 22 (flags:S; msg:"Potential SYN Flood on SSH Port"; threshold: type both, track by_src, count 20, seconds 5; sid:1000015; rev:1;)”
- We are sending SYN packets “ sudo hping3 -S -p 22 -i u1000 192.168.218.102”



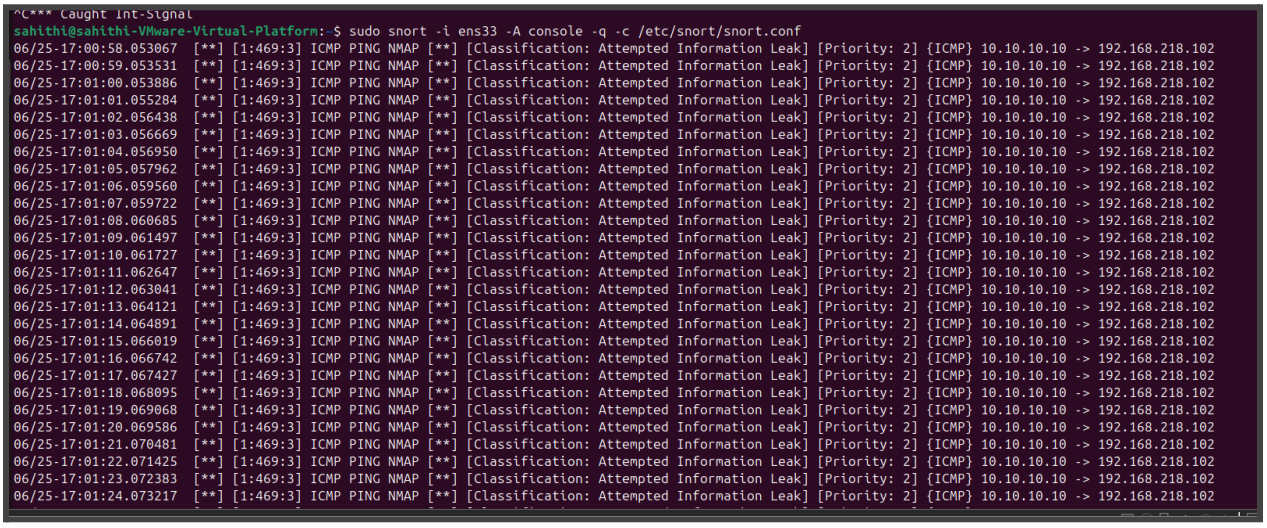
- Start running the snort “ sudo snort -i ens33 -A console -q -c /etc/snort/snort.conf”
- Alerts were triggered when 20+ SYN packets were detected within 5 seconds from the same source.
- TCP packets from **source port 0** (invalid/malformed).
- Traffic from **ports 20 (FTP)** and **53 (DNS)** targeting SSH—suspicious and unexpected.



5.4. Spoofed ICMP Echo Request

- To simulate a suspicious ICMP echo from a fake source, “ sudo hping3 -1 -a 10.10.10.10 192.168.218.102”
- In this -1 ICMP echo, -a is the spoofed source IP

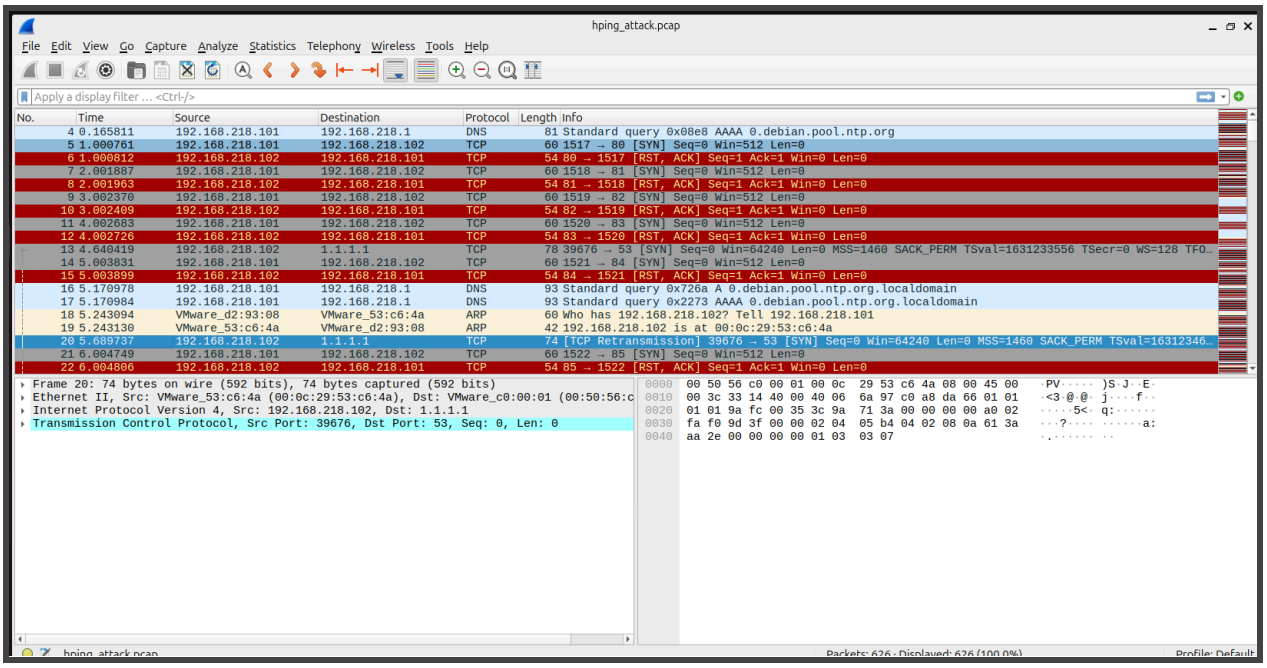
- An attacker system (10.10.10.10) ran to sent ICMP echo requests (ping scan) toward the target (192.168.218.102) for host discovery.
- Snort identified and logged multiple alerts using rule SID:469, labelled as “ICMP PING NMAP”, classified under Attempted Information Leak with Priority 2.



6. Traffic Capture & Log Analysis

6.1. Network capturer and analysis

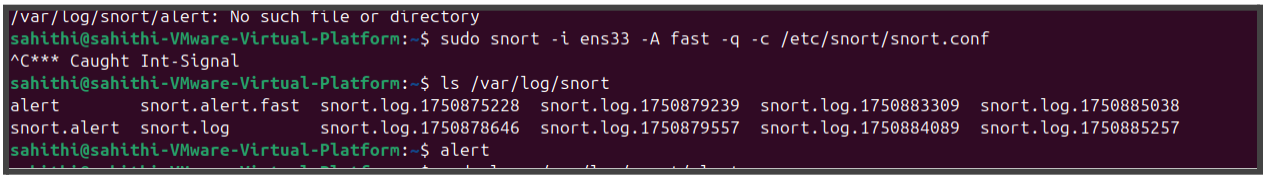
- Launch tcpdump and start monitoring the interface `ens33` and write the captured packets to a `.pcap` file “sudo tcpdump -i ens33 -w hping_attack.pcap”
- This captures all incoming packets—including potential SYN flood traffic—for offline analysis
- Run a simulated attack using `hping3` to send rapid TCP SYN packets “sudo hping3 -S -p ++50 192.168.218.102” We need to stop and save the files “ ls -lh hping_attack.pcap”
- Now open the Wireshark “wireshark hping_attack.pcap”



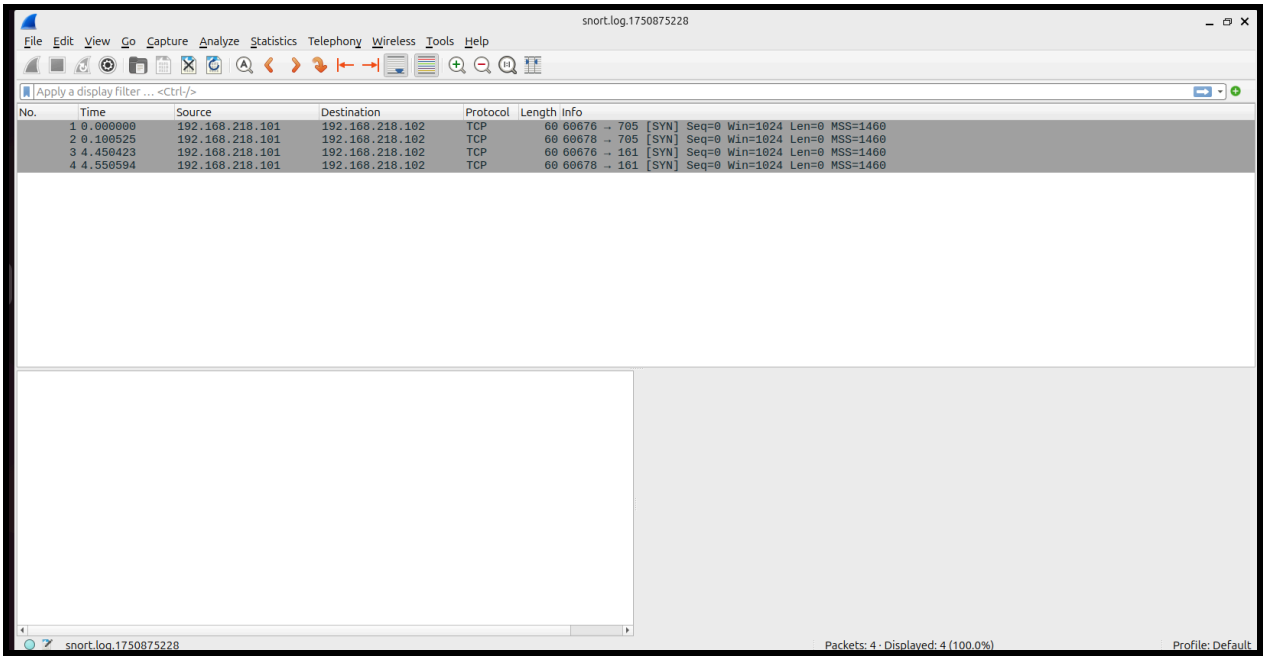
- Allows detailed packet analysis, including TCP flags, sequence numbers, and timing—helpful for confirming SYN flood patterns.
- The `.pcap` shows **TCP, DNS, and ARP** packets, indicating both normal background communication and possible attack traffic.

6.2. SYN Flood Log Analysis Using Snort and Wireshark

- Open the Snort log " sudo less /var/log/snort/alert"
- We need to verify the alert directory " ls /var/log/snort"

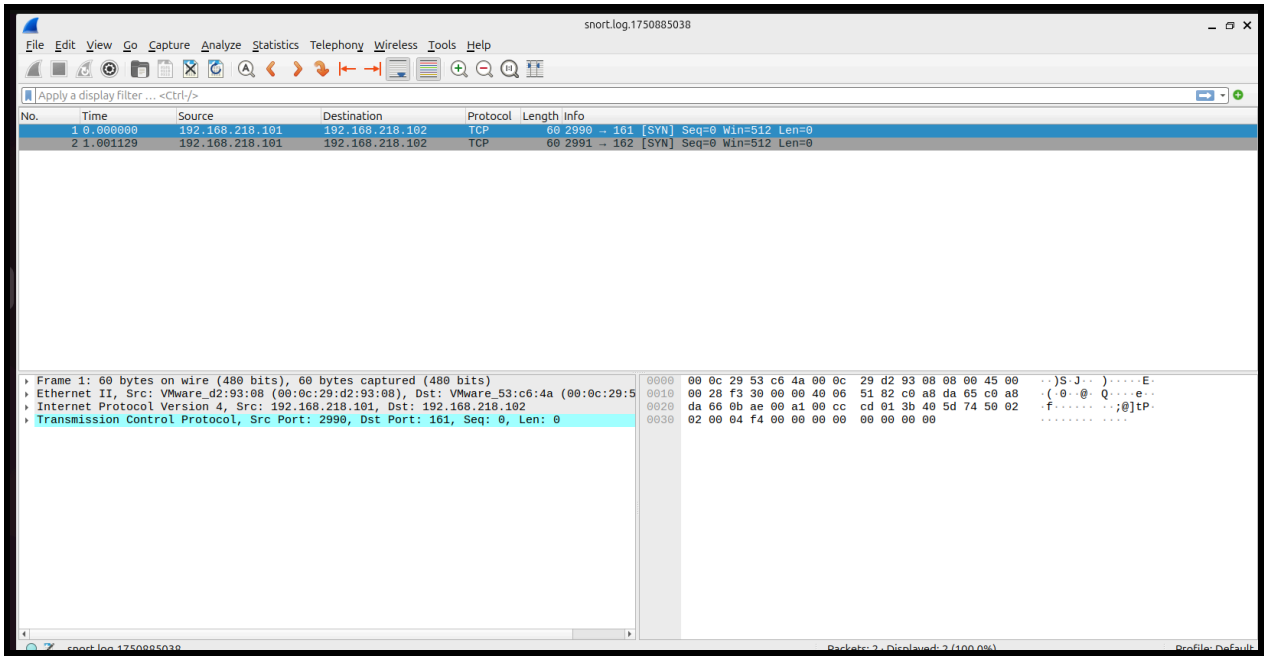


- Now we need "sudo less /var/log/snort/alert"
- wireshark /var/log/snort/snort.log.1750875228
- Permit me " sudo chmod 644 /var/log/snort/snort.log.1750875228", it helps to access the file.



- Multiple repeated **TCP SYN packets** with identical sequence numbers and no response (**no SYN-ACK**), typical of a **SYN flood**.
- Source: 192.168.218.101 → Destination: 192.168.218.102 on different ports (e.g., 705, 161).
- Now we need “sudo less /var/log/snort/alert”
- wireshark /var/log/snort/snort.log.1750885038
- Permit me “ sudo chmod 644 /var/log/snort/snort.log.1750885038” to access this file.

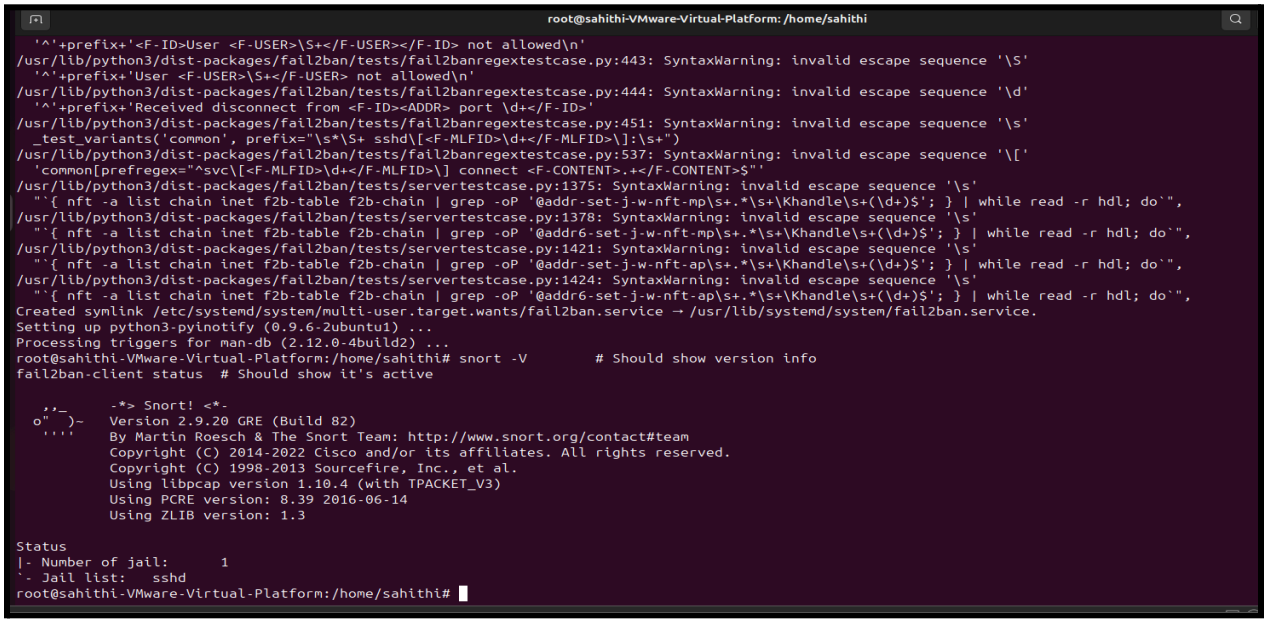
```
sahithi@sahithi-VMware-Virtual-Platform:~$ wireshark /var/log/snort/snort.log.1750885038
^C
sahithi@sahithi-VMware-Virtual-Platform:~$ sudo chmod 644 /var/log/snort/snort.log.1750885038
sahithi@sahithi-VMware-Virtual-Platform:~$ wireshark /var/log/snort/snort.log.1750885038
```



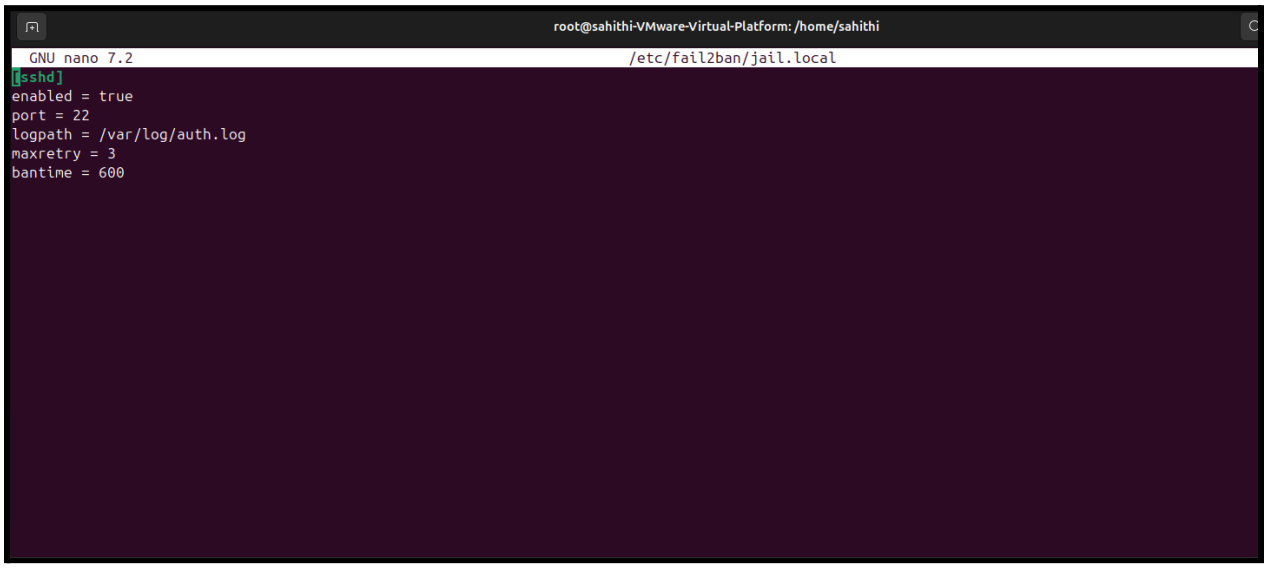
- This capture shows a **TCP SYN packet** sent from 192.168.218.101 (Src Port: 2990) to 192.168.218.102 (Dst Port: 161).
- The packet length is 60 bytes, and the **TCP flags field shows only SYN set**, which aligns with the pattern of a SYN flood

7. fail2ban Configuration & Jail Testing

- First, we need to install fail2ban “sudo apt install fail2ban”
- We need to verify the installation “ snort -V
  fail2ban-client status”



- Create SSH/jail configuration “ sudo nano /etc/fail2ban/jail.local”



- We need to test the jail from Kali “ ssh invaliduser@192.168.181.102”

```
┌──(root㉿kali)-[/home/kali]
└─# ssh invaliduser@192.168.181.102

The authenticity of host '192.168.181.102 (192.168.181.102)' can't be established.
ED25519 key fingerprint is SHA256:zNGDas/2746TZtCYqmSZxmSSVPK6M43oeY8BErM9TB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added '192.168.181.102' (ED25519) to the list of known hosts.
invaliduser@192.168.181.102's password:
Permission denied, please try again.
invaliduser@192.168.181.102's password:
Permission denied, please try again.
invaliduser@192.168.181.102's password:
Connection closed by 192.168.181.102 port 22

┌──(root㉿kali)-[/home/kali]
└─#
```

- After doing it two to three times, we need to check “sudo fail2ban-client status sshd”

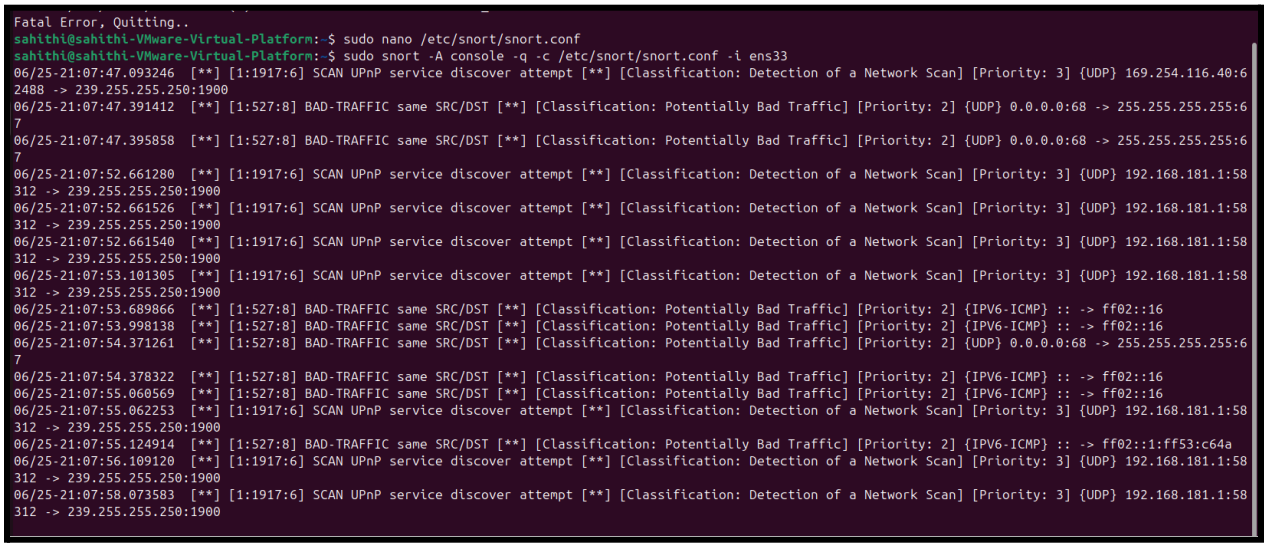
```
   `- Banned IP list:
root@sahithi-VMware-Virtual-Platform:/home/sahithi# sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     3
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   192.168.181.101
root@sahithi-VMware-Virtual-Platform:/home/sahithi#
```

- We need to remove the band on Kali “sudo fail2ban-client set sshd unbanip 192.168.181.101”

```
   |- Total banned:     1
   `- Banned IP list:   192.168.181.101
root@sahithi-VMware-Virtual-Platform:/home/sahithi# sudo fail2ban-client set sshd unbanip 192.168.181.101
1
root@sahithi-VMware-Virtual-Platform:/home/sahithi# sudo fail2ban-client set sshd unbanip 192.168.181.101
0
root@sahithi-VMware-Virtual-Platform:/home/sahithi#
```

8. Adding Custom Rules to Reduce False Positives

- First, we need to edit the rules “ sudo nano /etc/snort/rules/local.rules”
- Adding text rule to ICMP “ alert icmp any any -> $HOME_NET any (msg:"ICMP test alert"; sid:1000001; rev:1;)”
- Run Snort “ sudo snort -i ens33 -A console -q -c /etc/snort/snort.conf”
- Move to Kali and try running “ ping 192.168.181.102”



- This screenshot shows Snort actively logging alerts in console mode, detecting repeated TCP and UPnP traffic. These alerts illustrate typical false positives, later fine-tuned using suppression rules.

9. Observations

- The Snort IDS successfully detected a wide range of simulated network intrusions, including SYN scans, brute-force SSH attempts, SYN floods, and spoofed ICMP echo requests.
- The combined use of UFW, Snort, and fail2ban created a multi-layered security defence that monitored, alerted, and actively responded to threats in real time.
- Log analysis using `tcpdump`, `Wireshark`, and Snort’s own alert files provided insight into network behaviour and traffic anomalies.
- Custom rules and threshold tuning significantly reduced false positives, helping refine the detection accuracy and overall usability of the IDS/IPS setup.

10. Conclusion

This project demonstrates the effectiveness of a properly configured open-source intrusion detection and prevention system in a controlled lab environment. By integrating Snort, fail2ban, and UFW, the lab achieved comprehensive coverage for threat visibility, traffic control, and automated mitigation. The simulation of real-world attacks and the use of forensic tools like Wireshark validated the setup's capability to detect and respond to malicious behaviour.

With further optimisation and rule tuning, this solution could serve as a foundational security layer for small to medium-sized networks or as a learning environment for security professionals in training.