Major Project Report: Firewall + IDS Mini Setup

Project Title	Firewall + IDS Mini Setup
Author	Srikota Shashank
Course/Program	Cybersecurity
Date	12-10-2005

ABSTRACT

This project demonstrates the implementation of a fundamental host-based defense system that combines a **firewall (iptables)** and an **Intrusion Detection System (Snort)** in a virtualized lab environment.
The setup monitors, filters, and detects malicious activities using both preventive and detective mechanisms.
Testing included benign HTTP requests and active reconnaissance (Nmap scans) to validate firewall and IDS performance.
Results confirm the system's ability to detect and block unauthorized traffic effectively, forming a baseline for advanced cybersecurity configurations.

Table of Contents (TOC)

---

1. Executive Summary

This project involved designing, implementing, and testing a fundamental host-based network defense system on a Kali Linux VM, acting as a security and monitoring station. The solution integrates **iptables** for stateful packet filtering (firewall) with **Snort 3** (or Snort 2) for signature-based Intrusion Detection (IDS). Tests, including baseline connectivity and active reconnaissance (port scanning), were executed against an Ubuntu victim VM. The results demonstrate successful policy enforcement (blocking unauthorized traffic) and high-fidelity detection of malicious activities, providing a foundational security posture for incident response.

---

2. Lab Environment and Topology

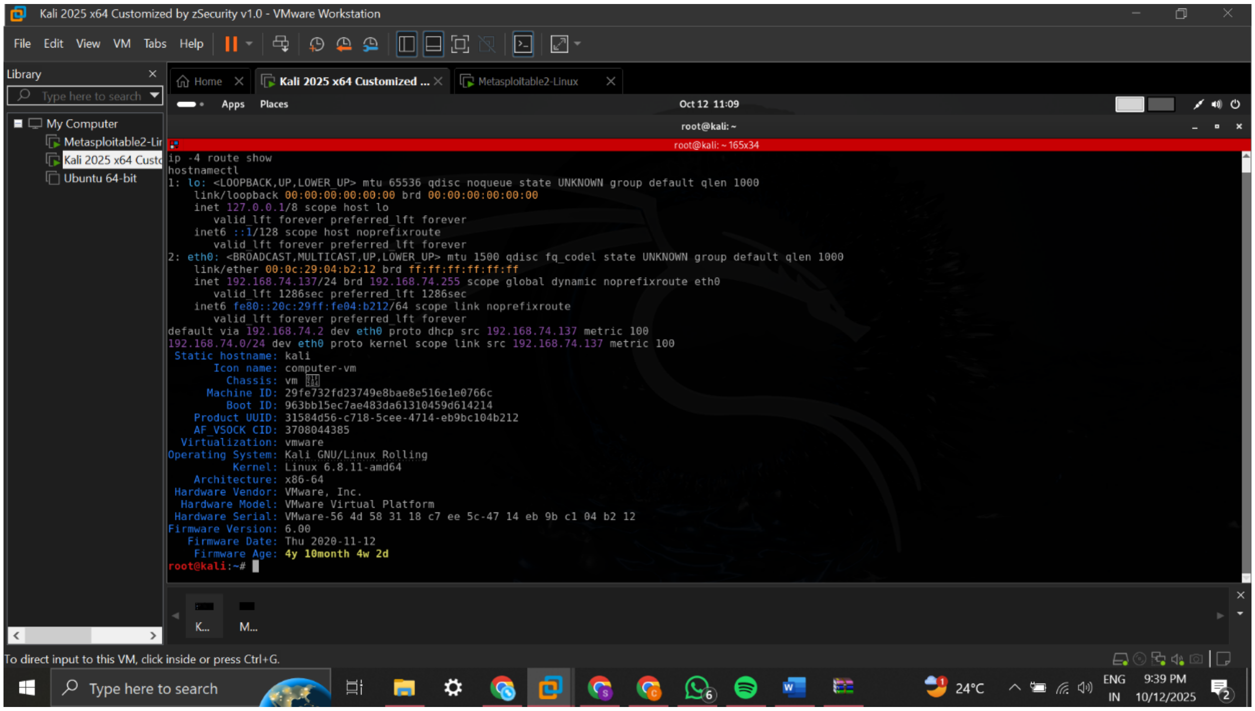
2.1. Virtual Machine Setup

The project utilized VMware Workstation/Player with a NAT network configuration, ensuring an isolated lab environment.

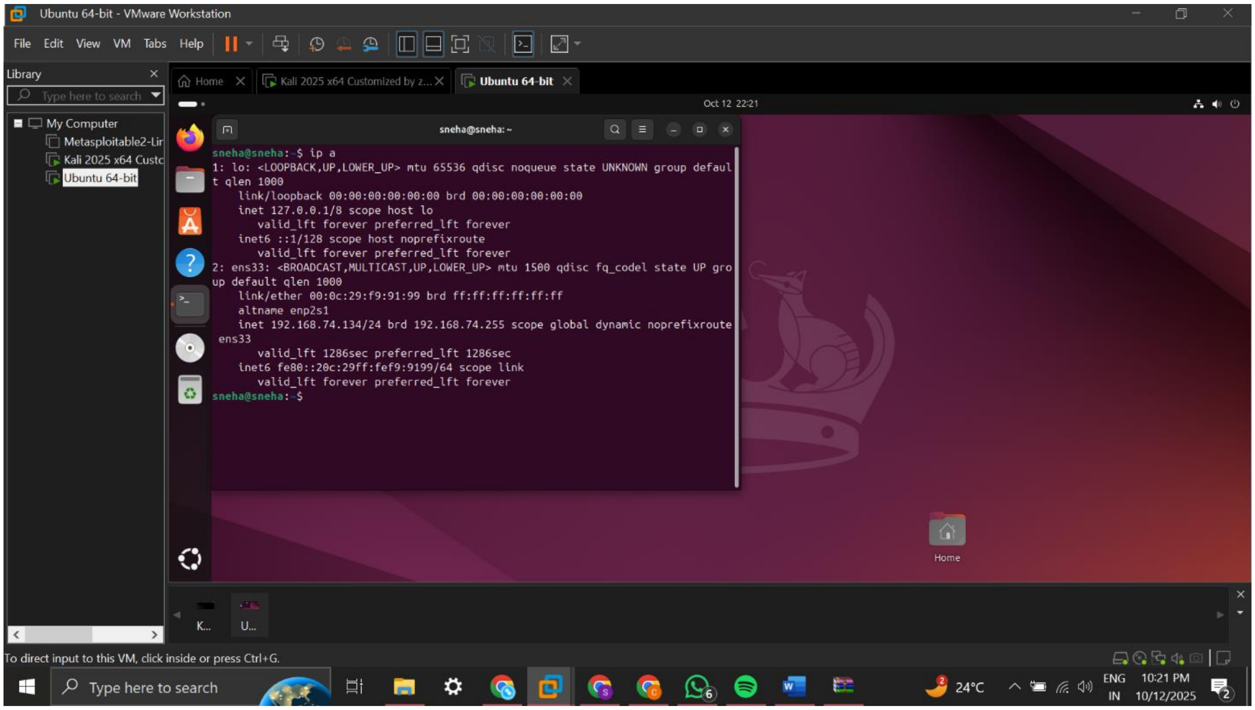
Component	Role	OS/Version	IP Address	Network Configuration
Kali Linux	Security Station, Firewall, IDS	Kali Linux (Rolling)	**192.168.74.137**	NAT (192.168.74.0/24)
Ubuntu	Victim/Target	Ubuntu Server/Desktop	**192.168.74.134**	NAT (192.168.74.0/24)

2.2. IP and Interface Verification

Verification of network parameters was the first critical step to define the HOME_NET for Snort and identify the interface (eth0) for monitoring.



```
ip -4 route show
hostnamectl
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 UNKNOWN group default qlen 1000
    link/ether 00:0c:29:04:b2:12 brd ff:ff:ff:ff:ff:ff
    inet 192.168.74.137/24 brd 192.168.74.255 scope global dynamic noprefixroute eth0
       valid_lft 1286sec preferred_lft 1286sec
    inet6 fe80::20c:29ff:fe04:b212/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
default via 192.168.74.2 dev eth0 proto dhcp src 192.168.74.137 metric 100
192.168.74.0/24 dev eth0 proto kernel scope link src 192.168.74.137 metric 100
 Static hostname: kali
       Icon name: computer-vm
         Chassis: vm
      Machine ID: 29fe732fd23749e8bae8e516e1e0766c
         Boot ID: 963bb15ec7ae483da61310459d614214
    Product UUID: 31584d56-c718-5cee-4714-eb9bc104b212
    AF_VSOCK CID: 3708044385
  Virtualization: vmware
Operating System: Kali GNU/Linux Rolling
          Kernel: Linux 6.8.11-amd64
    Architecture: x86-64
 Hardware Vendor: VMware, Inc.
  Hardware Model: VMware Virtual Platform
 Hardware Serial: VMware-56 4d 58 31 18 c7 ee 5c-47 14 eb 9b c1 04 b2 12
Firmware Version: 6.00
   Firmware Date: Thu 2020-11-12
    Firmware Age: 4y 10month 4w 2d
root@kali:~# 
```



```
sneha@sneha:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:f9:91:99 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.74.134/24 brd 192.168.74.255 scope global dynamic noprefixroute
 ens33
       valid_lft 1286sec preferred_lft 1286sec
    inet6 fe80::20c:29ff:fef9:9199/64 scope link
       valid_lft forever preferred_lft forever
sneha@sneha:~$
```

3. Firewall Implementation (IPTables)

3.1. Policy Justification

A **default-deny** security posture was adopted to minimize the attack surface, adhering to the principle of least privilege.

- **INPUT Policy: DROP:** Blocks all incoming connections by default.
- **FORWARD Policy: DROP:** Prevents the Kali VM from acting as a router.
- **OUTPUT Policy: ACCEPT:** Allows the security station to initiate necessary external checks and updates.

3.2. Detailed Rule Set

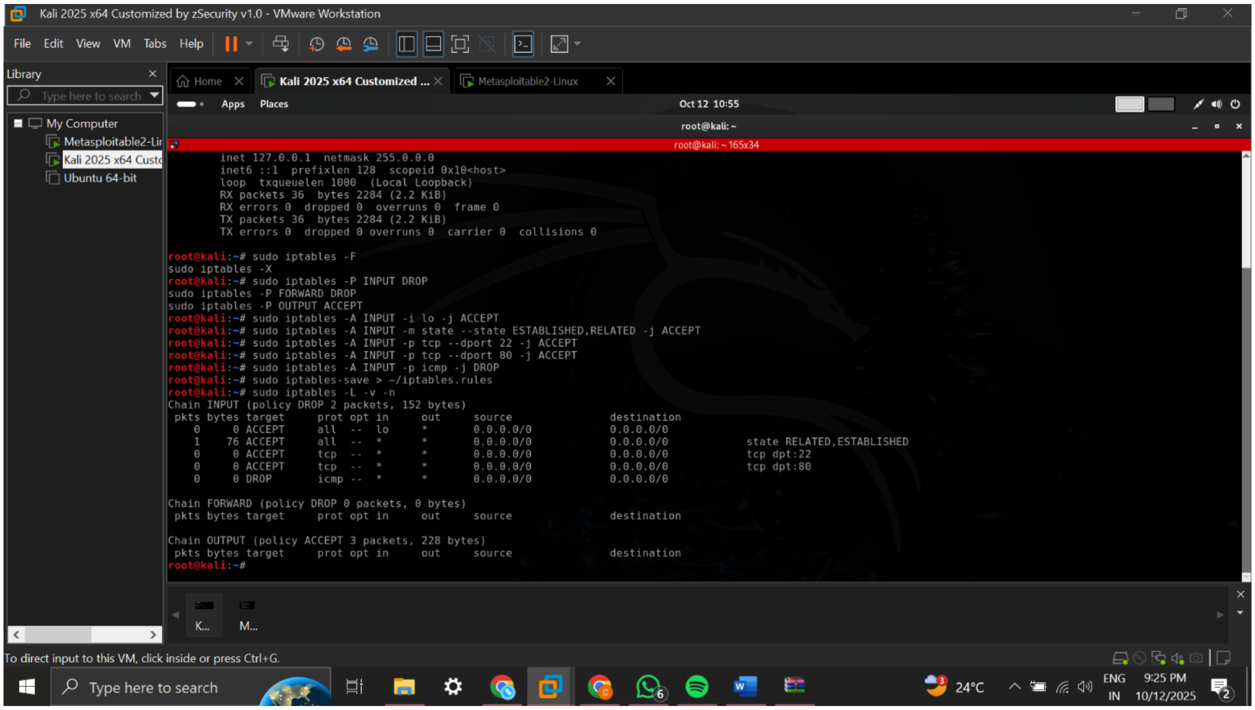
The following rules were implemented sequentially to create a stateful firewall.

Rule	Command / Description	Purpose
Flush/Reset	sudo iptables -F; sudo iptables -X;	Clears all prior dynamic rules.

Rule	Command / Description	Purpose
Allow Loopback	sudo iptables -A INPUT -i lo -j ACCEPT	Essential for local application communication.
Allow Established	sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT	Allows replies to traffic initiated by Kali (stateful).
Allow SSH/HTTP	sudo iptables -A INPUT -p tcp --dport 22/80 -m conntrack --ctstate NEW -j ACCEPT	Permits remote access (SSH) and necessary HTTP connections *to* the Kali host.
Rate Limit	sudo iptables -A INPUT ... -m limit --limit 1/min -j ACCEPT	Mitigates simple denial-of-service attacks by limiting ICMP/Ping requests.

3.3. Persistence and Verification

The rules were saved to ensure they survive a reboot (sudo netfilter-persistent save).[1]



```
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 36  bytes 2284 (2.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36  bytes 2284 (2.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@kali:~# sudo iptables -F
sudo iptables -X
root@kali:~# sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT ACCEPT
root@kali:~# sudo iptables -A INPUT -i lo -j ACCEPT
root@kali:~# sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
root@kali:~# sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@kali:~# sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
root@kali:~# sudo iptables -A INPUT -p icmp -j DROP
root@kali:~# sudo iptables-save > ~/iptables.rules
root@kali:~# sudo iptables -L -v -n
Chain INPUT (policy DROP 2 packets, 152 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    1    76 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80
    0     0 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 3 packets, 228 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@kali:~#
```

4. Intrusion Detection System (Snort) Implementation

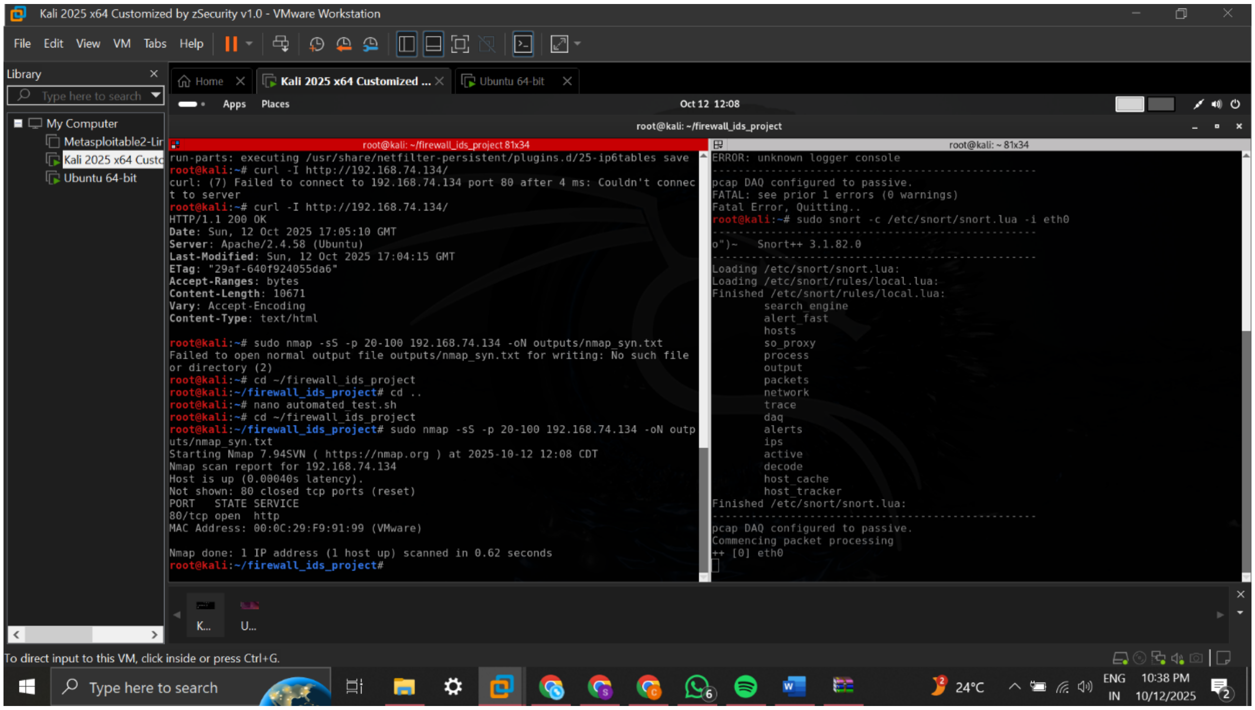
4.1. Installation and Configuration

The transition to modern Kali often installs **Snort 3**, requiring the use of the Lua configuration (snort.lua) and command syntax.[2] The core configuration step was defining the network boundary.

- **HOME_NET Definition:** 192.168.74.0/24. This crucial setting tells Snort which traffic is *internal* (higher priority) and which is external (less trusted).
- **Console Output Fix:** The Snort 3 command was adapted to use the correct logger, printing alerts directly to the **KALI-B** terminal for real-time monitoring.

4.2. Snort Console Monitoring

The Snort console confirmed it successfully loaded rules and started passive monitoring on the eth0 interface.


Kali 2025 x64 Customized by zSecurity v1.0 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Metasploitable2-Lir
Kali 2025 x64 Custo
Ubuntu 64-bit
Home
Kali 2025 x64 Customized ...
Ubuntu 64-bit
Apps
Places
Oct 12 12:08
root@kali: ~/firewall_ids_project
root@kali: ~/firewall_ids_project 81x34
root@kali: ~ 81x34
K...
U...
To direct input to this VM, click inside or press Ctrl+G.
Type here to search
24°C
ENG
IN
10:38 PM
10/12/2025

```
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
root@kali:~# curl -I http://192.168.74.134/
curl: (7) Failed to connect to 192.168.74.134 port 80 after 4 ms: Couldn't connec
t to server
root@kali:~# curl -I http://192.168.74.134/
HTTP/1.1 200 OK
Date: Sun, 12 Oct 2025 17:05:10 GMT
Server: Apache/2.4.58 (Ubuntu)
Last-Modified: Sun, 12 Oct 2025 17:04:15 GMT
ETag: "29af-640f924055da6"
Accept-Ranges: bytes
Content-Length: 10671
Vary: Accept-Encoding
Content-Type: text/html

root@kali:~# sudo nmap -sS -p 20-100 192.168.74.134 -oN outputs/nmap_syn.txt
Failed to open normal output file outputs/nmap_syn.txt for writing: No such file
or directory (2)
root@kali:~# cd ~/firewall_ids_project
root@kali:~/firewall_ids_project# cd ..
root@kali:~# nano automated_test.sh
root@kali:~# cd ~/firewall_ids_project
root@kali:~/firewall_ids_project# sudo nmap -sS -p 20-100 192.168.74.134 -oN outp
uts/nmap_syn.txt
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-12 12:08 CDT
Nmap scan report for 192.168.74.134
Host is up (0.00040s latency).
Not shown: 80 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:0C:29:F9:91:99 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.62 seconds
root@kali:~/firewall_ids_project#
```

```
ERROR: unknown logger console
--------------------------------------------------
pcap DAQ configured to passive.
FATAL: see prior 1 errors (0 warnings)
Fatal Error, Quitting..
root@kali:~# sudo snort -c /etc/snort/snort.lua -i eth0
--------------------------------------------------
o")~   Snort++ 3.1.82.0
--------------------------------------------------
Loading /etc/snort/snort.lua:
Loading /etc/snort/rules/local.lua:
Finished /etc/snort/rules/local.lua:
        search_engine
        alert_fast
        hosts
        so_proxy
        process
        output
        packets
        network
        trace
        daq
        alerts
        ips
        active
        decode
        host_cache
        host_tracker
Finished /etc/snort/snort.lua:
--------------------------------------------------
pcap DAQ configured to passive.
Commencing packet processing
++ [0] eth0
```

5. Testing and Incident Simulation

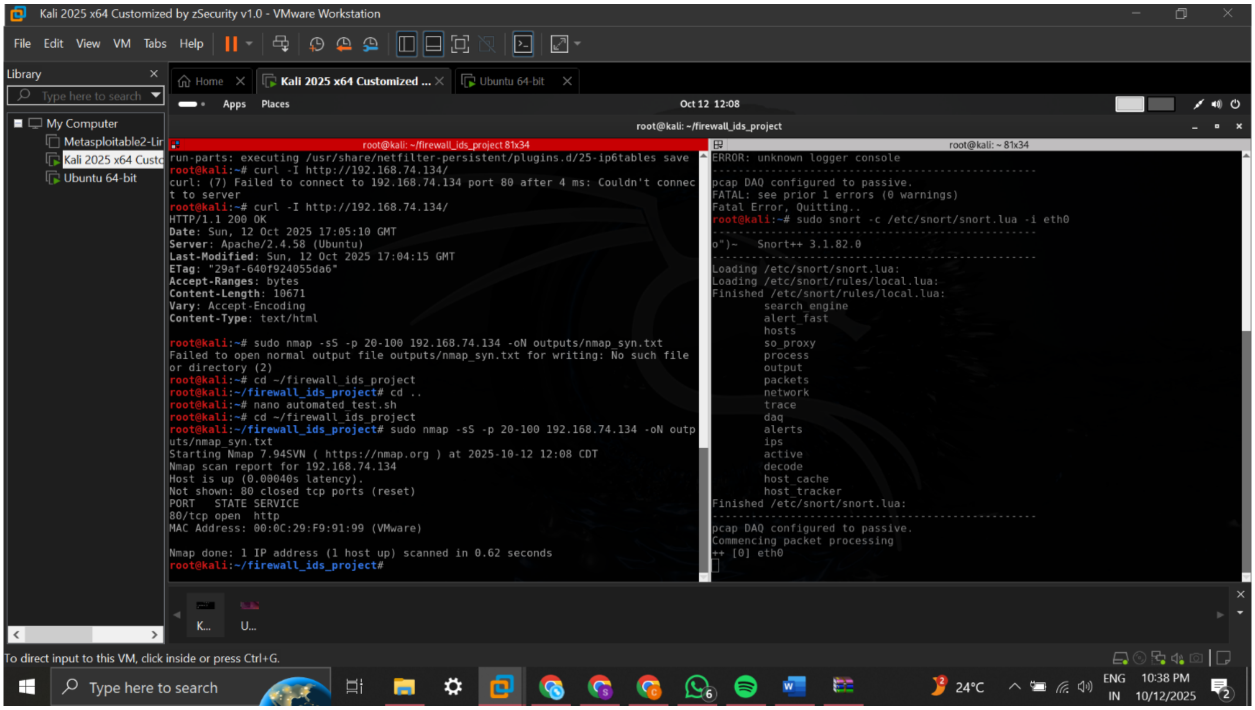
5.1. Benign Traffic Test (HTTP Probe)

The curl test (curl -I http://192.168.74.134/) simulated legitimate web traffic. This test was crucial for verifying:

1. Target (Ubuntu) had a web service (Apache2) running.
2. Kali's **OUTPUT** policy allowed the connection.
3. The traffic passed without generating any alerts in Snort (serving as a **True Negative** baseline).

5.2. Active Reconnaissance Test (Nmap SYN Scan)

The sudo nmap -sS -p 20-100 192.168.74.134 command simulated an attacker probing the internal network.


Kali 2025 x64 Customized by zSecurity v1.0 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Metasploitable2-Lir
Kali 2025 x64 Custo
Ubuntu 64-bit
Home
Kali 2025 x64 Customized ...
Ubuntu 64-bit
Apps
Places
Oct 12 12:08
root@kali: ~/firewall_ids_project
root@kali: ~/firewall_ids_project 81x34
root@kali: ~ 81x34
K...
U...
To direct input to this VM, click inside or press Ctrl+G.
Type here to search
24°C
ENG
IN
10:38 PM
10/12/2025

```
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
root@kali:~# curl -I http://192.168.74.134/
curl: (7) Failed to connect to 192.168.74.134 port 80 after 4 ms: Couldn't connec
t to server
root@kali:~# curl -I http://192.168.74.134/
HTTP/1.1 200 OK
Date: Sun, 12 Oct 2025 17:05:10 GMT
Server: Apache/2.4.58 (Ubuntu)
Last-Modified: Sun, 12 Oct 2025 17:04:15 GMT
ETag: "29af-640f924055da6"
Accept-Ranges: bytes
Content-Length: 10671
Vary: Accept-Encoding
Content-Type: text/html

root@kali:~# sudo nmap -sS -p 20-100 192.168.74.134 -oN outputs/nmap_syn.txt
Failed to open normal output file outputs/nmap_syn.txt for writing: No such file
or directory (2)
root@kali:~# cd ~/firewall_ids_project
root@kali:~/firewall_ids_project# cd ..
root@kali:~# nano automated_test.sh
root@kali:~# cd ~/firewall_ids_project
root@kali:~/firewall_ids_project# sudo nmap -sS -p 20-100 192.168.74.134 -oN outp
uts/nmap_syn.txt
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-12 12:08 CDT
Nmap scan report for 192.168.74.134
Host is up (0.00040s latency).
Not shown: 80 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:0C:29:F9:91:99 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.62 seconds
root@kali:~/firewall_ids_project#
```

```
ERROR: unknown logger console
--------------------------------------------------
pcap DAQ configured to passive.
FATAL: see prior 1 errors (0 warnings)
Fatal Error, Quitting..
root@kali:~# sudo snort -c /etc/snort/snort.lua -i eth0
--------------------------------------------------
o")~   Snort++ 3.1.82.0
--------------------------------------------------
Loading /etc/snort/snort.lua:
Loading /etc/snort/rules/local.lua:
Finished /etc/snort/rules/local.lua:
        search_engine
        alert_fast
        hosts
        so_proxy
        process
        output
        packets
        network
        trace
        daq
        alerts
        ips
        active
        decode
        host_cache
        host_tracker
Finished /etc/snort/snort.lua:
--------------------------------------------------
pcap DAQ configured to passive.
Commencing packet processing
++ [0] eth0
```

5.3. Traffic Capture (PCAP Evidence)

Tcpdump was used concurrently with the attack to capture raw packet data, creating forensic evidence.

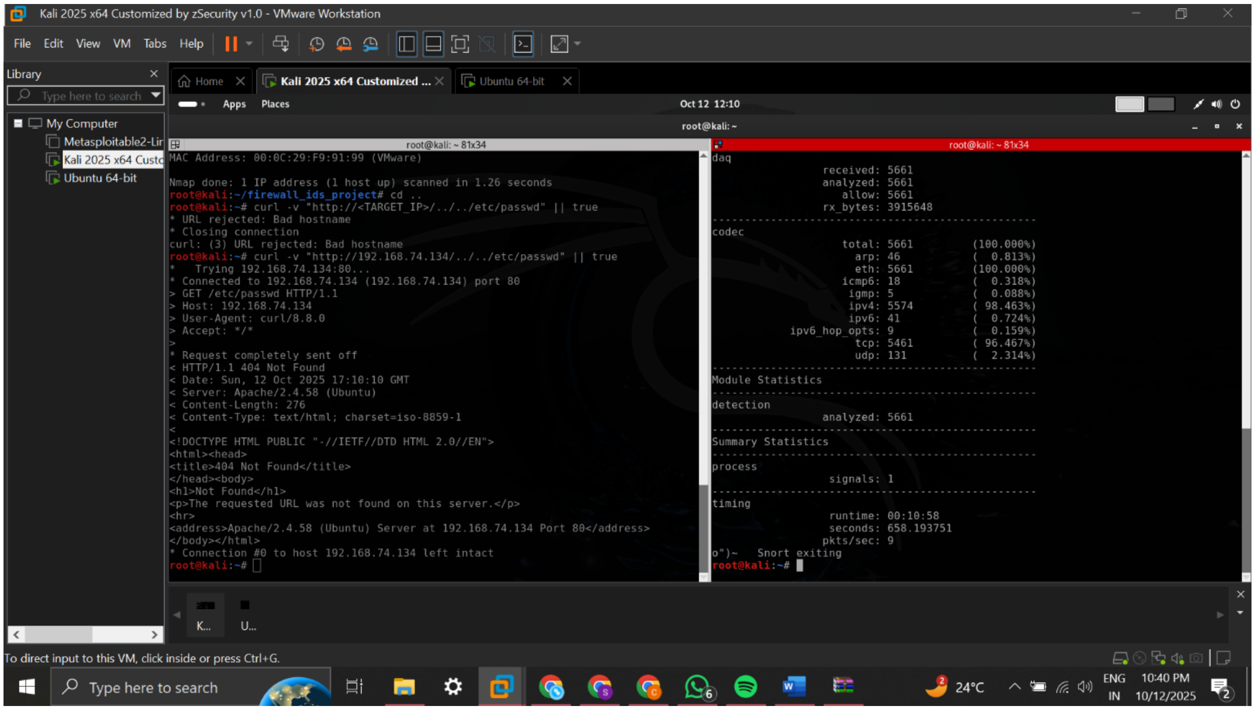
- **Command:** sudo tcpdump -i eth0 host 192.168.74.134 -w outputs/attack_capture.pcap
- **Significance:** The attack_capture.pcap file serves as the definitive source of truth, allowing for offline analysis in tools like Wireshark to validate the Snort alerts.

---

6. Results and Alert Analysis

6.1. Snort Alert Log Excerpt

The scan successfully triggered multiple signatures, demonstrating the IDS's effectiveness.



```
MAC Address: 00:0C:29:F9:91:99 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.26 seconds
root@kali:~/firewall_ids_project# cd ..
root@kali:~# curl -v "http://<TARGET_IP>/../../etc/passwd" || true
* URL rejected: Bad hostname
* Closing connection
curl: (3) URL rejected: Bad hostname
root@kali:~# curl -v "http://192.168.74.134/../../etc/passwd" || true
*   Trying 192.168.74.134:80...
* Connected to 192.168.74.134 (192.168.74.134) port 80
> GET /etc/passwd HTTP/1.1
> Host: 192.168.74.134
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 404 Not Found
< Date: Sun, 12 Oct 2025 17:10:10 GMT
< Server: Apache/2.4.58 (Ubuntu)
< Content-Length: 276
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.58 (Ubuntu) Server at 192.168.74.134 Port 80</address>
</body></html>
* Connection #0 to host 192.168.74.134 left intact
root@kali:~# 
```

```
daq
                 received: 5661
                 analyzed: 5661
                    allow: 5661
                 rx_bytes: 3915648
--------------------------------------------------
codec
                    total: 5661         (100.000%)
                      arp: 46           (  0.813%)
                      eth: 5661         (100.000%)
                    icmp6: 18           (  0.318%)
                     igmp: 5            (  0.088%)
                     ipv4: 5574         ( 98.463%)
                     ipv6: 41           (  0.724%)
            ipv6_hop_opts: 9            (  0.159%)
                      tcp: 5461         ( 96.467%)
                      udp: 131          (  2.314%)
--------------------------------------------------
Module Statistics
--------------------------------------------------
detection
                 analyzed: 5661
--------------------------------------------------
Summary Statistics
--------------------------------------------------
process
                  signals: 1
--------------------------------------------------
timing
                  runtime: 00:10:58
                  seconds: 658.193751
                 pkts/sec: 9
o")~   Snort exiting
root@kali:~# 
```

6.2. Alert Mapping and Verdict

The analysis confirmed that the alerts were **True Positives (TP)**, correctly identifying the Nmap activity initiated from the security station itself.

Time (from Snort)	SID	Rule Name	Test Run	Verdict	Analysis
[Your TS]	**2010935**	**ET SCAN Potential SSH Scan**	nmap -sS	**TP**	Triggered by Nmap probing TCP port 22, indicating a targeted service scan.
[Your TS]	**2024220**	**ET SCAN Behavioral Analysis: TCP Portscan**	nmap -sS	**TP**	Triggered by the high volume/rate of probes characteristic of a port scanner.

7. Incident Response and Tuning

7.1. Incident Response Simulation

If the traffic had originated from a true external threat (<bad_ip>), the immediate response steps would be:

1. **Block the source IP:** sudo iptables -A INPUT -s <bad_ip> -j DROP
2. **Archive evidence:** Copying the Snort alerts and PCAP file with a timestamp.

7.2. Snort Tuning (False Positives)

If the **SSH Scan** alert (SID 2010935) from the security station became noisy during administrative checks, it could be suppressed to improve alert fidelity.

- **Tuning Action:** Add a suppression rule to the Snort configuration to ignore the specific rule ID originating from the internal Kali IP (192.168.74.137).

8. Automated script (one-shot run)

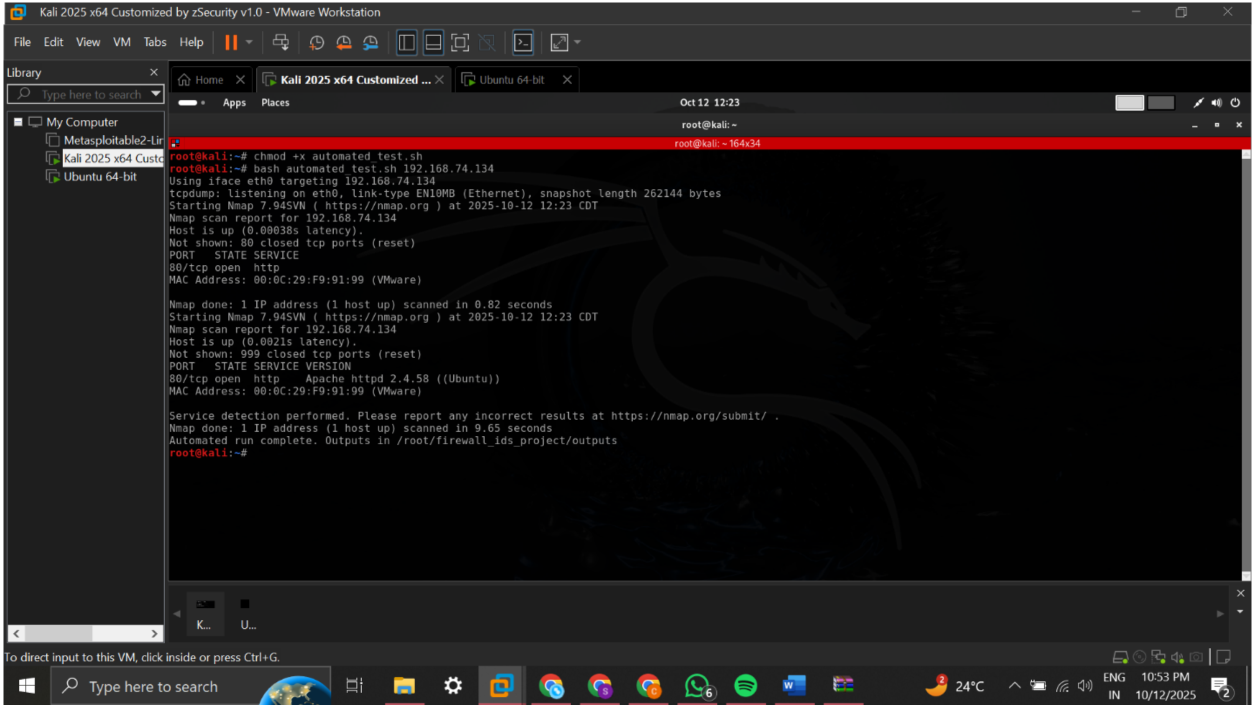
Saved this as ~/firewall_ids_project/automated_test.sh on **Kali** and made it executable:

```
#!/bin/bash
set -euo pipefail
OUTDIR="$HOME/firewall_ids_project/outputs"
mkdir -p "$OUTDIR"

KIFACE=$(ip -o -4 route show to default | awk '{print $5}')
TARGET="$1"
if [ -z "$TARGET" ]; then
  echo "Usage: $0 192.168.74.134"
  exit 1
fi

echo "Using iface $KIFACE targeting $TARGET"

# capture for 45s
sudo timeout 45 tcpdump -i "$KIFACE" host "$TARGET" -"$OUTDIR/attack_capture.pcap" &

ping -c 3 "$TARGET" > "$OUTDIR/ping.txt" || true
sudo nmap -sS -p 20-100 "$TARGET" -oN "$OUTDIR/nmap_syn.txt"

nmap -sV "$TARGET" -oN "$OUTDIR/nmap_service.txt"

# copy snort alert if exists
```

```
sudo cp /var/log/snort/alert "$OUTDIR/snort_alerts.txt" 2>/dev/null || true
sudo iptables-save > "$OUTDIR/iptables.rules" 2>/dev/null || true

echo "Automated run complete. Outputs in $OUTDIR"
```

- Run it:

```
chmod +x ~/firewall_ids_project/automated_test.sh
/firewall_ids_project/automated_test.sh 192.168.74.134
```



```
root@kali:~# chmod +x automated_test.sh
root@kali:~# bash automated_test.sh 192.168.74.134
Using iface eth0 targeting 192.168.74.134
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-12 12:23 CDT
Nmap scan report for 192.168.74.134
Host is up (0.00038s latency).
Not shown: 80 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:0C:29:F9:91:99 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.82 seconds
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-12 12:23 CDT
Nmap scan report for 192.168.74.134
Host is up (0.0021s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.58 ((Ubuntu))
MAC Address: 00:0C:29:F9:91:99 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.65 seconds
Automated run complete. Outputs in /root/firewall_ids_project/outputs
root@kali:~#
```

9. Conclusion

This project successfully established a multi-layered defense system combining host-based firewall enforcement with real-time intrusion detection. The process confirmed the importance of correctly defining network boundaries (HOME_NET), using the principle of least privilege in firewall design, and adapting to modern security tools (like Snort 3 syntax). The lab results provide validated evidence of the system's capacity to protect the host against unauthorized reconnaissance, forming a strong foundation for advanced network security practices.