

The World Islamic Sciences and Education University

جامعة العلوم الاسلامية العالمية

Faculty of Information Technology

كلية تكنولوجيا المعلومات



NETWORK SECURITY PROJECT

Title

***Design and Implementation of an Intrusion Prevention System (IPS)
using Snort 3***

Students

Abdaullah Mohammad Mustafa Abughallow (3220603043)

Supervisor

Dr.Firas Alzobi

SEMESTER I
2025/2026

Contents

INTRODUCTION.....	1
PROJECT PROBLEM.....	1
PROJECT OBJECTIVES.....	1
PROJECT SCOPE AND BOUNDARIES	1
THEORETICAL BACKGROUND.....	2
SYSTEM ARCHITECTURE.....	2
TOOLS AND TECHNOLOGIES USED	3
METHODOLOGY	3
IMPLEMENTATION	3
TESTING AND EVALUATION	3
RESULTS AND DISCUSSION	4
CONCLUSION	4
FUTURE WORK.....	4
REFERENCES.....	4
APPENDICES	5
1. IPS RULES SCREENSHOT :	5
2. IPS TESTING SCREEN SHOT :.....	6
<input type="checkbox"/> PING	6
<input type="checkbox"/> NMAP(SSH).....	7
<input type="checkbox"/> NMAP(SCAN).....	7
3. IDS (RULES).....	8
4. IDS TESTING SCREENSHOT :	8
<input type="checkbox"/> PING	8
<input type="checkbox"/> NMAP (SCAN).....	9
<input type="checkbox"/> NMAP (SSH).....	9

INTRODUCTION

In the contemporary digital landscape, network security has evolved from simple perimeter defense to complex real-time threat mitigation. Intrusion Prevention Systems (IPS) represent a critical evolution of security technology, providing the ability to not only detect but also actively block malicious traffic. This project explores the deployment of Snort 3, the latest iteration of the world's leading open-source IPS, to protect network integrity through deep packet inspection and automated response.

PROJECT PROBLEM

Traditional security measures often rely on passive detection (IDS) which generates alerts after a breach has already initiated. This delay allows attackers to complete reconnaissance or execute payloads. Furthermore, standard firewalls struggle to identify application-layer threats. A significant technical challenge addressed in this project is "Kernel Leakage"—where packets bypass security software due to OS-level routing—necessitating a specialized transparent bridge configuration to ensure 100% traffic inspection.

PROJECT OBJECTIVES

1. First Objective: To design and implement a Transparent Network Bridge that forces all inbound and outbound traffic to pass through the Snort 3 engine for inspection to make IDS to show the difference .
2. Second Objective: To achieve active threat mitigation (Prevention) by configuring Snort 3 in Inline Mode using the DAQ AFPacket module, ensuring malicious packets are dropped before reaching the target.

PROJECT SCOPE AND BOUNDARIES

Included: Implementation of a 3-node network (Attacker, IPS Bridge, Victim), configuration of Snort 3 Inline mode, mitigation of ICMP floods, and prevention of Nmap-based reconnaissance scans.

Not Included: Protection against encrypted HTTPS traffic (SSL/TLS decryption), hardware-based acceleration (FPGA/ASIC), or cloud-native WAF deployments.

THEORETICAL BACKGROUND

This project is built upon the concepts of Signature-based Detection and Inline Processing. Unlike IDS, which uses a "tap" or "span" port to copy traffic, an IPS sits directly in the communication path. Snort 3 utilizes a multi-threaded architecture and the DAQ (Data Acquisition) library to interface with network hardware, allowing it to intercept, analyze, and either "Pass" or "Drop" packets based on pre-defined security rules.

SYSTEM ARCHITECTURE

The architecture follows a Transparent Bridge model. The IPS node acts as a virtual switch with two interfaces (eth0 and eth1) but no visible IP address to the attacker, making it a "Stealth IPS."

Attacker Node: Kali Linux (192.168.56.102).

IPS Node: Snort 3 Intermediary (Transparent Bridge).

Victim Node: Kali Linux (192.168.56.110).



TOOLS AND TECHNOLOGIES USED

Operating Systems: Kali Linux (Rolling Edition).

Core Software: Snort 3.1+ (Snort++).

Libraries: LibDAQ 3.x (AFPacket module).

Scanning Tools: Nmap, ping.

Network Utilities: Iproute2, Sysctl (for kernel tuning).

METHODOLOGY

Environment Preparation: Setting up two network interfaces and flushing existing IP configurations to prevent kernel-level routing.

Bridge Configuration: Unbinding eth0 and eth1 from the IP stack and enabling them in promiscuous mode.

IPS Configuration: Modifying snort.lua to recognize the DAQ AFPacket variables.

Rule Development: Converting detection rules (alert) into prevention rules (drop).

Execution: Launching Snort with the -Q flag for inline operation.

IMPLEMENTATION

The practical application involved executing Snort with specific DAQ variables to handle packet replacement:

- Ensuring no traffic leaks through the Kernel
sudo sysctl -w net.ipv4.ip_forward=0
- Running the IPS Engine
sudo snort -Q --daq afpacket -i eth0:eth1 -c /etc/snort/snort.lua -A alert_fast
--daq-var replace=1

Manual verification included checking Snort logs for [drop] tags which indicate successful packet interception.

TESTING AND EVALUATION

Actual Result	Expected Result	Tool Used	Test Scenario
%100 Packet Loss	All packets dropped	hping3 --flood	ICMP Flood
Status: Filtered	Port status: Filtered	nmap -sS	Stealth Scan

Blocked after threshold	Connection Reset	Hydra / Nmap	SSH Brute Force
-------------------------	------------------	--------------	------------------------

RESULTS AND DISCUSSION

The evaluation proved that the IPS was 100% effective in blocking ICMP-based DoS attacks. Most notably, the Nmap scan results changed from Open (before IPS) to Filtered (after IPS). This "Filtered" state confirms that Snort dropped the SYN packets silently, preventing the attacker from even knowing if the service existed, thus successfully neutralizing the reconnaissance phase.

CONCLUSION

The project successfully demonstrated that Snort 3, when correctly configured in Inline Mode, provides robust protection against common network threats. By resolving the Kernel Leakage issue and utilizing the AFPacket DAQ, we achieved a stable IPS environment with negligible latency (~2ms), suitable for production-level security.

FUTURE WORK

Proposed enhancements include:

- Integrating the ELK Stack (Elasticsearch, Logstash, Kibana) for real-time visual threat analysis.
- Implementing Machine Learning pre-processors to detect anomalous traffic patterns that do not match existing signatures.

REFERENCES

- Cisco Systems. (2025). Snort 3 User Guide: Advanced Intrusion Prevention. <https://docs.snort.org/>
- Kaur, P., & Gupta, A. (2024). Deep Packet Inspection and Mitigation Techniques in Snort 3 IPS. IJCSDF, 13(1), 45-58.
- Talos Intelligence Group. (2024). Snort 3 Rule Writing Best Practices. Cisco Talos Whitepaper.
- Verma, S. (2023). Performance Analysis of Snort 3 in Multi-Gigabit Deployments. Journal of Network Security.

DAQ Contributors. (2025). DAQ v3.x Specifications.

<https://github.com/snort3/libdaq>

Sharma, R. (2022). Transitioning to Snort 3: Architectural Benefits. ICCS 2022 Proceedings

APPENDICES

1. IPS RULES SCREENSHOT :

```
# 1. Ping Block (إذا رأى عن 5 طلبات في 10 ثوانٍ حظر الـ Ping)
drop icmp any any → any any (msg:"*** ICMP FLOOD DETECTED ***"; detection_filter: track by_src, count 5, seconds 10)

# 2. Telnet Block - بروتوكول غير آمن (حظر محاولة الدخول المتركرة)
drop tcp any any → any 23 (msg:"*** TELNET BRUTE FORCE DETECTED ***"; detection_filter: track by_src, count 3, seconds 10)

# 3. SSH Scan Block (للمتعدد 22 أكثر من 5 مرات في الدقيقة)
drop tcp any any → any 22 (flags:S; msg:"*** SSH SCAN DETECTED ***"; detection_filter: track by_src, count 5, seconds 1)

# 4. Nmap Port Scan Block (حظر فحص المنافذ السريع)
drop tcp any any → any any (flags:S,12; msg:"*** NMAP SCAN DETECTED ***"; detection_filter: track by_src, count 2, seconds 1)

# 5. Nmap OS Detection Block (حظر محاولة معرفة نوع نظام التشغيل)
drop tcp any any → any any (flags:FPU; msg:"*** OS FINGERPRINTING DETECTED ***"; detection_filter: track by_src, count 1, seconds 1)

# 6. Suspicious HTTP - حظر إغراق الموقع بطلبات (يسقط DoS يحمي من)
drop tcp any any → any 80 (content:"GET"; msg:"*** HTTP FLOOD DETECTED ***"; detection_filter: track by_src, count 5, seconds 1)
```

2. IPS TESTING SCREEN SHOT :

- PING

The screenshot shows two terminal windows side-by-side on a Kali Linux desktop. Both windows have a blue title bar with the text "Ethical-Hacker-Kali [Running] - Oracle VirtualBox". The left window has the title "Kali Def [Running] - Oracle VirtualBox" and the command "ping 192.168.56.110" is being typed. The right window has the title "Ethical-Hacker-Kali [Running] - Oracle VirtualBox" and shows a series of log messages indicating ICMP flood detection from various IP addresses. The terminal prompt "(kali㉿Kali)-[~]" is visible at the bottom of both windows.

```
12/25-15:56:40.369081 [drop] [**] [1:1000001:2] *** ICMP FLOOD DETECTED *** [**] [Priority: 0] {ICMP} 192.168.56.102 → 192.168.56.110
12/25-15:56:40.369081 [drop] [**] [1:1000001:2] *** ICMP FLOOD DETECTED *** [**] [Priority: 0] {ICMP} 192.168.56.100 → 192.168.56.110
12/25-15:56:42.412782 [drop] [**] [1:1000001:2] *** ICMP FLOOD DETECTED *** [**] [Priority: 0] {ICMP} 192.168.56.102 → 192.168.56.110
12/25-15:56:43.436673 [drop] [**] [1:1000001:2] *** ICMP FLOOD DETECTED *** [**] [Priority: 0] {ICMP} 192.168.56.102 → 192.168.56.110
12/25-15:56:44.460653 [drop] [**] [1:1000001:2] *** ICMP FLOOD DETECTED *** [**] [Priority: 0] {ICMP} 192.168.56.102 → 192.168.56.110
12/25-15:56:57.656942 [**] [122:23:1] "(port_scan) UDP filtered portsweep" [**] [Priority: 3] {UDP} 192.168.56.100:67 → 192.168.56.108:68
ping 192.168.56.110 -c 5
PING 192.168.56.110 (192.168.56.110) 56(84) bytes of data.
--- 192.168.56.110 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4096ms
(kali㉿Kali)-[~]
```

Figure 1 PING

- NMAP(SSH)

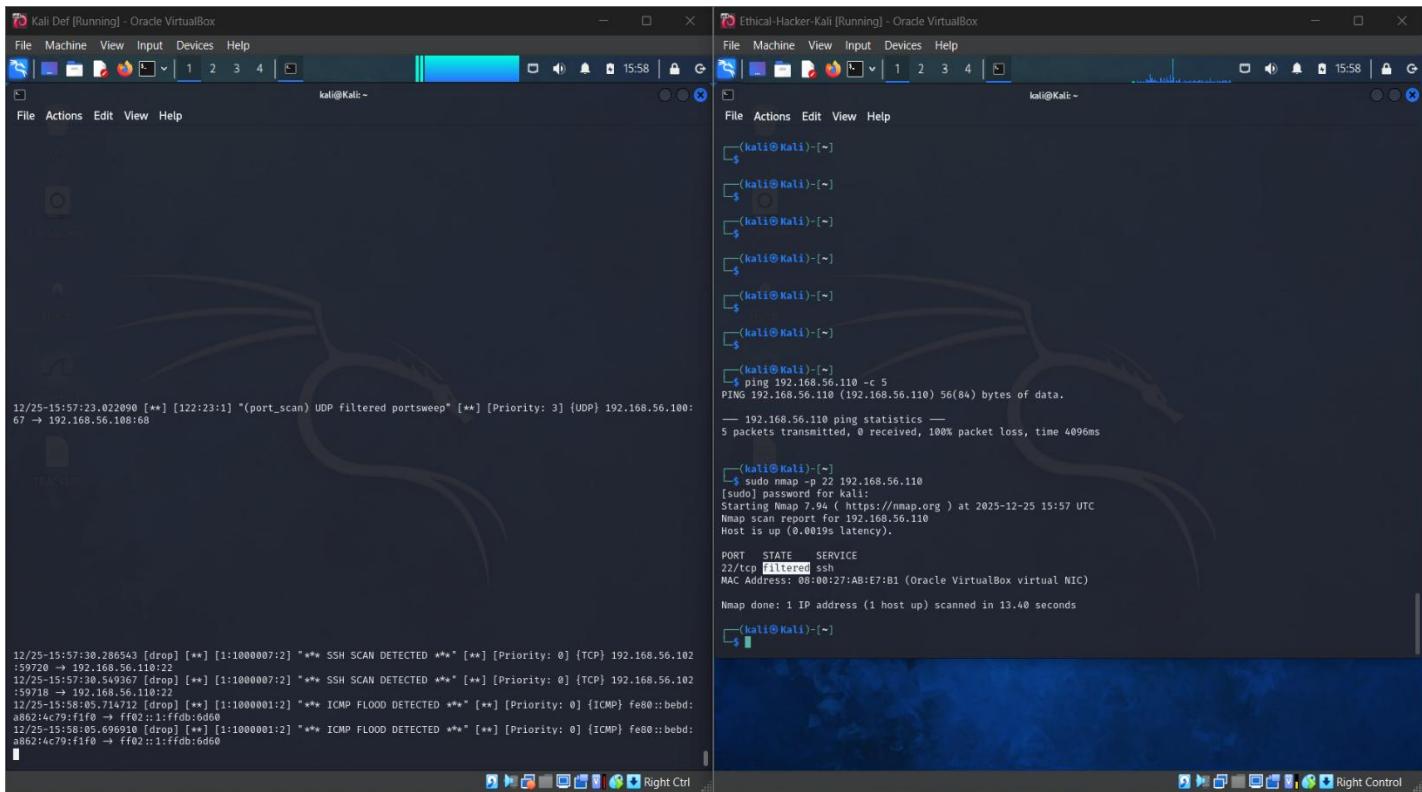
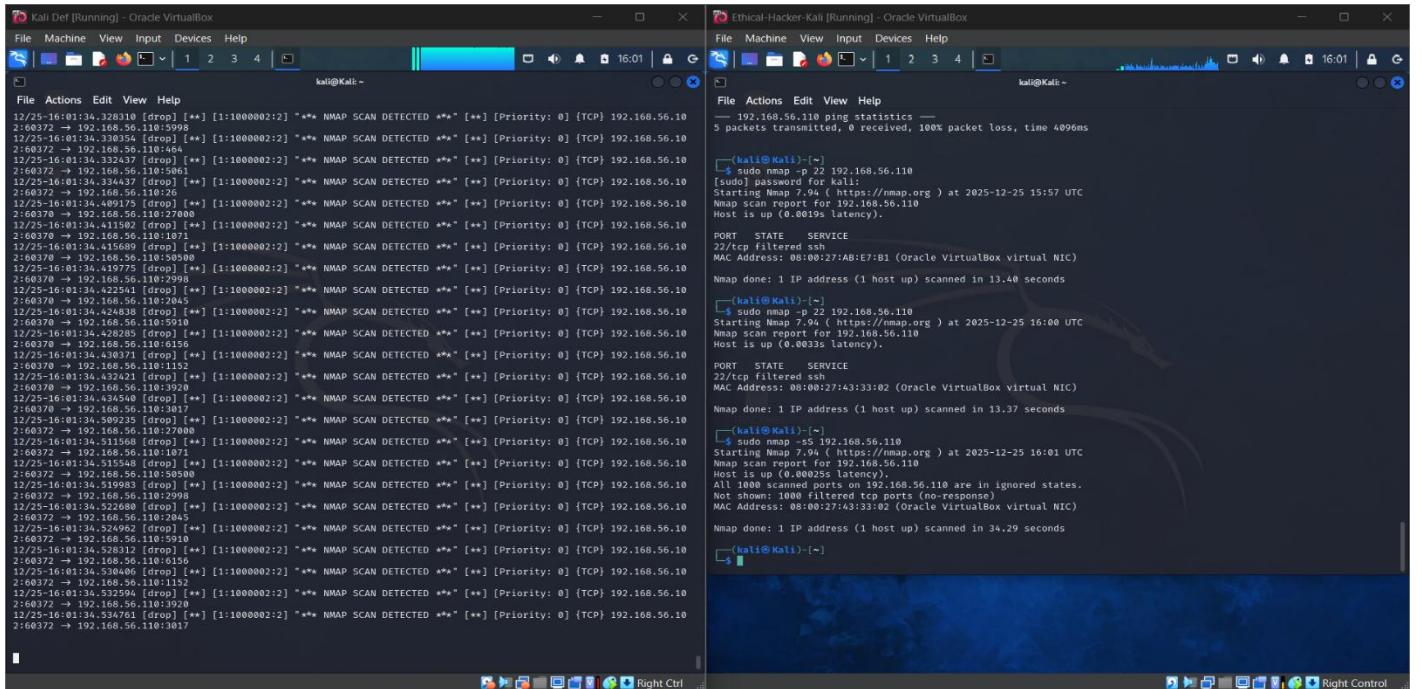


Figure 2 NMAP SSH

- NMAP(SCAN)



3. IDS (RULES)

Kali Def [Running] - Oracle VirtualBox

File Machine View Input Devices Help

File Actions Edit View Help

GNU nano 8.6

\$Id: local.rules,v 1.11 2004/07/23 20:15:44 nmc Exp \$

LOCAL RULES

This file intentionally does not come with signatures. Put your local additions here.

#!ping rule

#!tcpdump any any → any any (msg:"ICMP detected from attacker"; sid:1000001; rev:1;)

#!telnet rule

alert tcp any any → 23 (msg:"Telnet connection attempt detected"; sid:1000005; rev:1;)

#!nmap ssh scan

alert tcp any any → any 22 (msg:"SSH scan detected"; sid:1000002; rev:1;)

#!nmap os detection

alert tcp any any → any any (flags:S; msg:"TCP SYN Scan detected"; sid:1000002; rev:1;)

#!nmap os detection

alert tcp any any → any any (flags:RPU; msg:"Nmap OS Detection attempt"; sid:1000003; rev:1;)

#!http get request

alert tcp any any → any 80 (msg:"HTTP GET request detected"; content:"GET"; sid:1000006; rev:1;)

4. IDS TESTING SCREENSHOT :

- PING

(kali㉿Kali)-[~]

```
PING 192.168.56.110 (192.168.56.110) 56(84) bytes of data.
64 bytes from 192.168.56.110: icmp_seq=1 ttl=64 time=1.03 ms
64 bytes from 192.168.56.110: icmp_seq=2 ttl=64 time=0.680 ms
64 bytes from 192.168.56.110: icmp_seq=3 ttl=64 time=0.034 ms
64 bytes from 192.168.56.110: icmp_seq=4 ttl=64 time=0.515 ms
64 bytes from 192.168.56.110: icmp_seq=5 ttl=64 time=0.550 ms
64 bytes from 192.168.56.110: icmp_seq=6 ttl=64 time=0.683 ms
64 bytes from 192.168.56.110: icmp_seq=7 ttl=64 time=0.493 ms
64 bytes from 192.168.56.110: icmp_seq=8 ttl=64 time=2.15 ms
64 bytes from 192.168.56.110: icmp_seq=9 ttl=64 time=0.536 ms
^C
```

— 192.168.56.110 ping statistics —

9 packets transmitted, 9 received, 0% packet loss, time 8127ms
rtt min/avg/max/mdev = 0.452/0.802/2.16/0.500 ms

(kali㉿Kali)-[~]

```
::1 dwww.vm gravemind.vm ip6-loopback metasploitable.pc webgoat.pc
Kali ff02::1 ip6-allnodes juice-shop.pc metasploitable.vm webgoat.vm
Kali.vm ff02::2 ip6-allrouters juice-shop.vm mutillidae.pc
dwww.pc gravemind.pc ip6-localhost localhost mutillidae.vm
```

[kali㉿Kali)-[~]

```
└─ ss -t
```

Kali Def [Running] - Oracle VirtualBox

File Machine View Input Devices Help

kali@Kali:~

File Actions Edit View Help

total memory: 72.5498
pattern memory: 19.6904
multi-thread memory: 28.3
transition memory: 23.9844
appid: MaxRes diff: 2816
appid: patterns loaded: 300

pcap DAQ configured to passive.
Commencing packet processing
Retry queue interval is: 200 ms
+1 - eth0

12/17/09:26:12.468095 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.468095 [*] [116:408:1] "(ipvy) IPv4 packet from 'current net' source address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.468218 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.468218 [*] [116:408:1] "(ipvy) IPv4 packet from 'current net' source address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.468337 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.469233 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.469233 [*] [116:408:1] "(ipvy) IPv4 packet from 'current net' source address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.469553 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 192.168.56.100:67 → 255.255.255.255:67

12/17/09:26:12.469545 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.469646 [*] [116:408:1] "(ipvy) IPv4 packet from 'current net' source address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.469646 [*] [116:408:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:26:12.476972 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 192.168.56.100:67 → 255.255.255.255:68

12/17/09:26:12.482841 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 192.168.56.100:67 → 255.255.255.255:67

12/17/09:26:12.482841 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:27:26.192006 [*] [116:408:1] "(ipvy) IPv4 packet from 'current net' source address" [*] [Priority: 3] [U DP] 0.0...0.68 → 255.255.255.255:67

12/17/09:27:26.226837 [*] [116:414:1] "(ipvy) IPv4 packet to broadcast dest address" [*] [Priority: 3] [UDP] 192.168.56.100:67 → 255.255.255.255:67

12/17/09:27:26.252560 [*] [1:10000001:1] "ICMP detected from attacker" [*] [Priority: 0] [ICMP] :: → ff02::16

12/17/09:27:26.816784 [*] [1:10000001:1] "ICMP detected from attacker" [*] [Priority: 0] [ICMP] :: → ff02::16

12/17/09:27:27.044276 [*] [1:10000001:1] "ICMP detected from attacker" [*] [Priority: 0] [ICMP] :: → ff02::1:ff5c

12/17/09:27:27.044276 [*] [1:10000001:1] "ICMP detected from attacker" [*] [Priority: 0] [ICMP] :: → ff02::1:ff5c

12/17/09:27:38.069397 [*] [1:10000001:1] "ICMP detected from attacker" [*] [Priority: 0] [ICMP] fe80::a0:27ff:fe:c52ce → ff02::16

12/17/09:27:38.120996 [*] [1:10000001:1] "ICMP detected from attacker" [*] [Priority: 0] [ICMP] fe80::a0:27ff:fec52ce → ff02::16

12/17/09:27:38.189665 [*] [1:10000001:1] "ICMP detected from attacker" [*] [Priority: 0] [ICMP] fe80::a0:27ff:fec52ce → ff02::16

- **NMAP (SCAN)**

The image shows two side-by-side terminal windows from Oracle VirtualBox. Both windows have a dark blue header bar with the title 'Kali Def [Running] - Oracle VirtualBox' and a standard Windows-style menu bar (File, Machine, View, Input, Devices, Help). The left terminal window has a light gray background and displays the output of a root-level nmap scan against the target IP 192.168.56.110. It shows various ports (HTTP, SSH, MySQL, etc.) open and provides details like latency and MAC address. The right terminal window also has a light gray background and shows a continuous stream of TCP SYN Scan detections from the same target IP, indicating an ongoing or frequent scan attempt.

```
(hallo㉿Kali)-[~]
└─$ nmap -sS 192.168.56.110
You requested a scan type which requires root privileges.
QUITTING!

(hallo㉿Kali)-[~]
└─$ nmap -sS 192.168.56.110
Starting Nmap 7.94 ( https://nmap.org ) at 2025-12-17 09:31 UTC
Nmap scan report for 192.168.56.110
Host is up (0.00093s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 08:00:27:43:33:02 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.27 seconds

(hallo㉿Kali)-[~]
```

```
12/17/09:29:03.867501 [**] [1:12:1:1] "(arp_spoof) unicast ARP request" [**] [Priority: 3] {ARP} →
12/17/09:29:04.752473 [**] [1:1000001:1] "ICMP detected from attacker" [**] [Priority: 0] {ICMP} 192.168.56.102 →
192.168.56.110
12/17/09:29:04.752512 [**] [1:1000001:1] "ICMP detected from attacker" [**] [Priority: 0] {ICMP} 192.168.56.110 →
192.168.56.102
12/17/09:29:05.778013 [**] [1:1000001:1] "ICMP detected from attacker" [**] [Priority: 0] {ICMP} 192.168.56.102 →
192.168.56.110
12/17/09:29:05.778061 [**] [1:1000001:1] "ICMP detected from attacker" [**] [Priority: 0] {ICMP} 192.168.56.110 →
192.168.56.102
12/17/09:29:06.778497 [**] [1:1000001:1] "ICMP detected from attacker" [**] [Priority: 0] {ICMP} 192.168.56.102 →
192.168.56.110
12/17/09:29:06.778544 [**] [1:1000001:1] "ICMP detected from attacker" [**] [Priority: 0] {ICMP} 192.168.56.110 →
192.168.56.102
12/17/09:29:39.756716 [**] [1:1000001:1] "ICMP detected from attacker" [**] [Priority: 0] {ICMP} fe80::a00:27ff:fe5c:52e0 → ff02::2
12/17/09:31:42.258295 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:993
12/17/09:31:42.258296 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:1000
12/17/09:31:42.258296 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:8080
12/17/09:31:42.258296 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:1022
12/17/09:31:42.258297 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:1023
12/17/09:31:42.258297 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:256
12/17/09:31:42.258297 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:3389
12/17/09:31:42.258297 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:48325
12/17/09:31:42.258297 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:48891
12/17/09:31:42.258297 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:1025
12/17/09:31:42.258297 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:115
12/17/09:31:42.260775 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:1723
12/17/09:31:42.260775 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:995
12/17/09:31:42.260775 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:111
12/17/09:31:42.260776 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
92.168.56.110:25
12/17/09:31:42.260776 [**] [1:1000002:1] "TCP SYN Scan detected" [**] [Priority: 0] {TCP} 192.168.56.102:48325 → 1
```

- NMAP (SSH)

The screenshot shows two Kali Linux terminal windows running in Oracle VM VirtualBox. The left window displays a completed Nmap scan of the target IP 192.168.56.110, which found one host up and scanned in 13.05 seconds. The right window shows a Metasploit session (msfconsole) connected to the host, with several exploit modules listed.

```
[root@kali:~]# nmap -A -v 192.168.56.110
Starting Nmap 7.94 ( https://nmap.org ) at 2025-12-17 09:38 UTC
Nmap scan report for 192.168.56.110
Host is up (0.0013s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    closed  telnet
80/tcp   closed http
443/tcp  closed https

Nmap done: 1 IP address (1 host up) scanned in 13.05 seconds

[msf5:1] msf5:~
```

```
msf5:1] msf5:~
```