# ****Proof of Concept (PoC) – Network Intrusion Detection System (NIDS)****

**Prepared by:** Ishan Chowdhury  
**Intern ID:** 159  
**Team:** SkullFaced

## ****1. Objective****

This PoC demonstrates the design and testing of a lightweight **Network Intrusion Detection System (NIDS)**. The goal is to monitor network traffic (live or from PCAP files) and raise alerts for:

ICMP pings (echo request/reply)

TCP connection attempts (SYNs, half-open connections)

Port scanning techniques (SYN, NULL, FIN scans)

Suspicious behaviors such as floods or high-rate scans

## ****2. Lab Setup****

**Attacker System:** Kali Linux (10.12.67.74)

**Victim System:** Windows 10 VM ( 10.12.67.178)

**Host Machine:** Windows 11 with Python environment

**Network Mode:** Host-only or Bridged network

**Tools Required:**

Python 3.9+

Scapy library

Nmap (for scan generation)

hping3 (for flood simulation)

Wireshark (for PCAP validation, optional)

## ****3. Installation****

### ****Step 1 – Install Python and Scapy****

# On Windows host

pip install scapy

### ****Step 2 – Install Npcap on Windows****

Download and install Npcap.

## ****4. NIDS Implementation (Python Script)****

Save as nids.py

from scapy.all import sniff, IP, TCP, ICMP

from collections import defaultdict

icmp\_count = defaultdict(int)

syn\_count = defaultdict(int)

def process\_packet(pkt):

    if IP in pkt:

        src = pkt[IP].src

        dst = pkt[IP].dst

        # ICMP Detection

        if ICMP in pkt:

            icmp\_count[src] += 1

            if pkt[ICMP].type in [0, 8]:

                print(f"[ICMP] Ping detected from {src} to {dst}")

            if icmp\_count[src] > 5:

                print(f"[ALERT] ICMP flood detected from {src}")

        # TCP Scans

        if TCP in pkt:

            flags = pkt[TCP].flags

            if flags == 0x02:  # SYN

                syn\_count[src] += 1

                print(f"[TCP] SYN from {src} to {dst}:{pkt[TCP].dport}")

                if syn\_count[src] > 10:

                    print(f"[ALERT] High-rate SYN scan from {src}")

            elif flags == 0x00:  # NULL scan

                print(f"[TCP] NULL scan from {src} to {dst}")

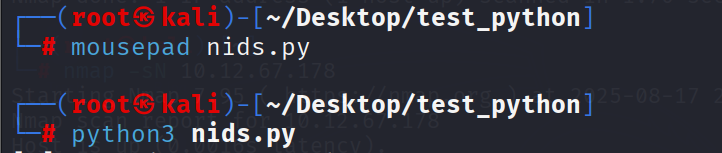
            elif flags == 0x01:  # FIN scan

                print(f"[TCP] FIN scan from {src} to {dst}")

print("[\*] Starting NIDS packet capture...")

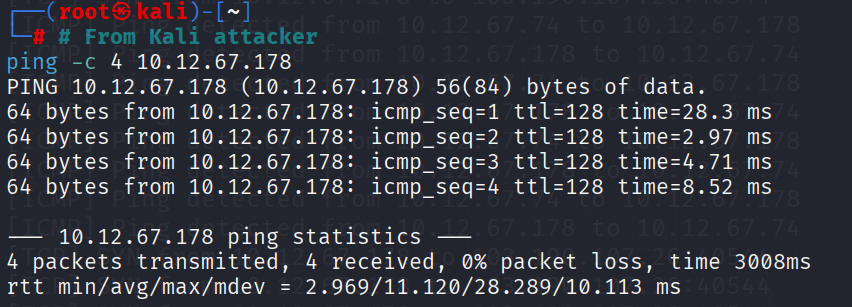
sniff(filter="ip", prn=process\_packet, store=0)

Run it with:

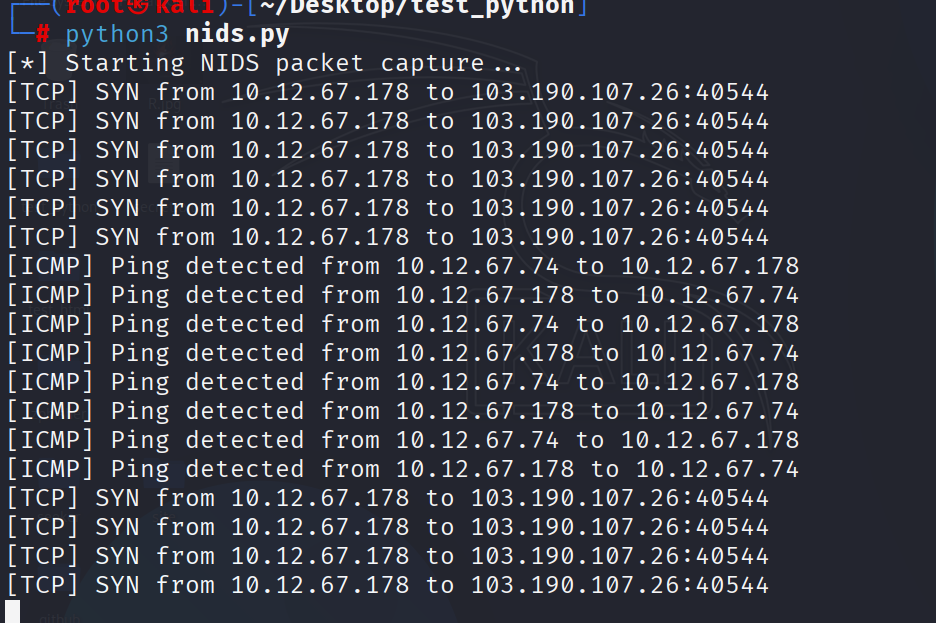


## ****5. Testing Procedures****

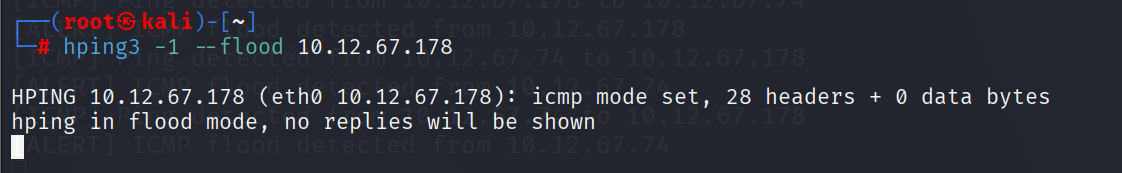
### ****Test 1 – ICMP Ping Detection****



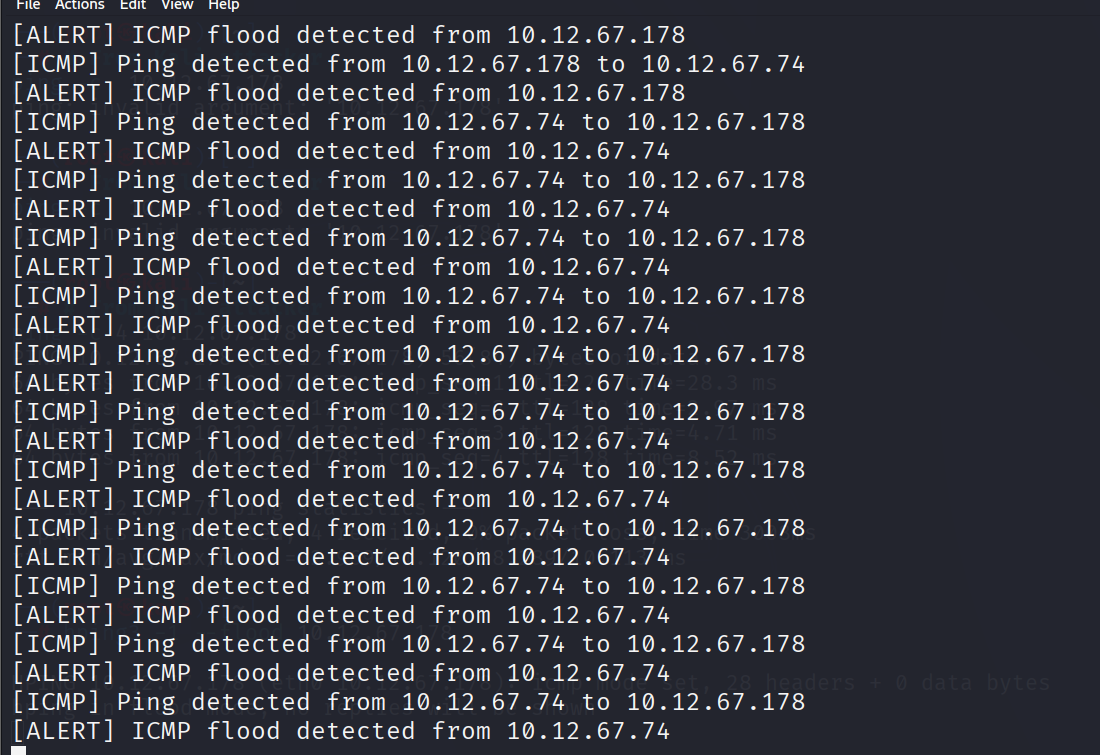
Expected: NIDS prints ICMP ping alerts.



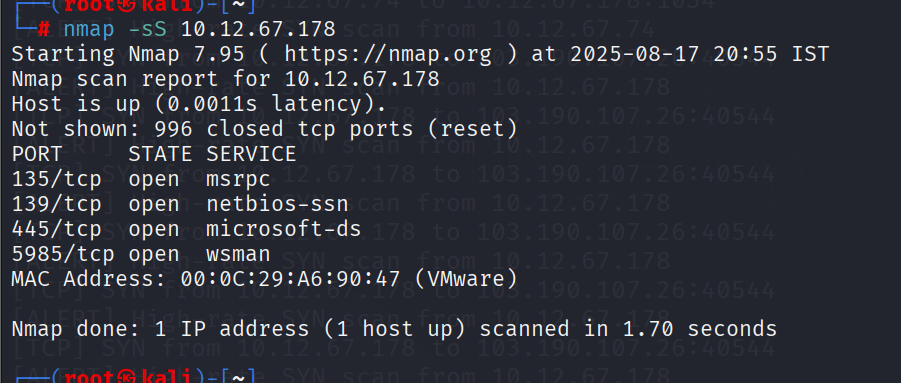
Test 2 – ICMP Flood Simulation



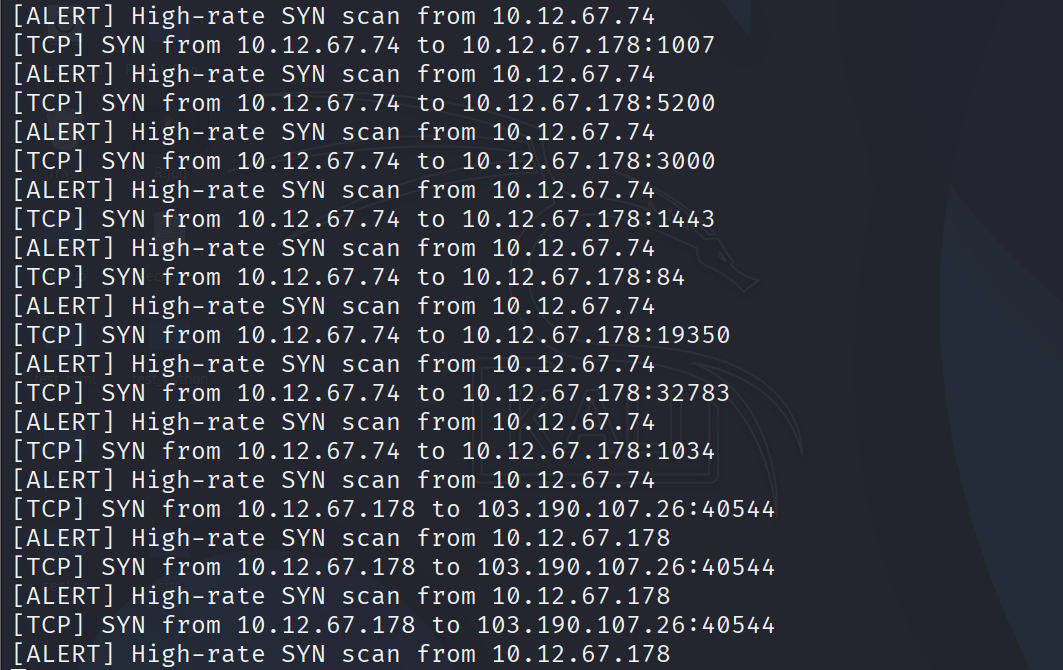
Expected: NIDS raises **ICMP flood alert**.



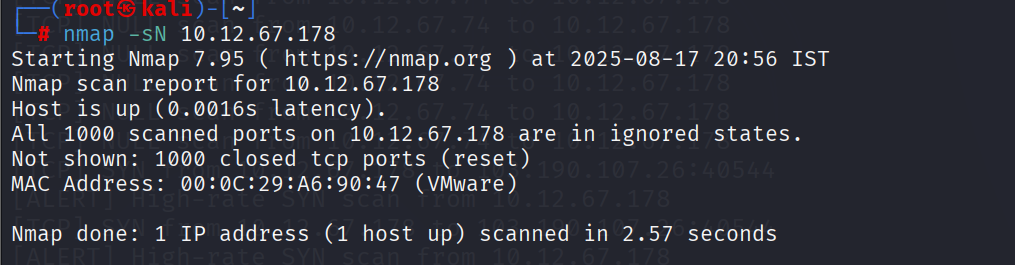
Test 3 – TCP SYN Scan



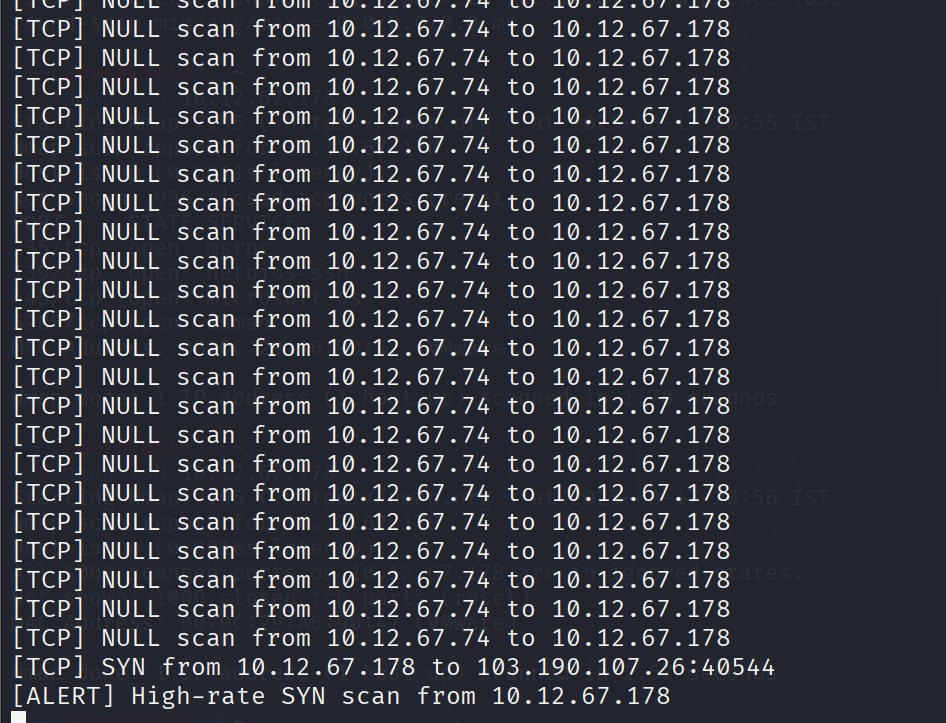
Expected: NIDS prints SYN connection attempts and detects **high-rate SYN scan**.



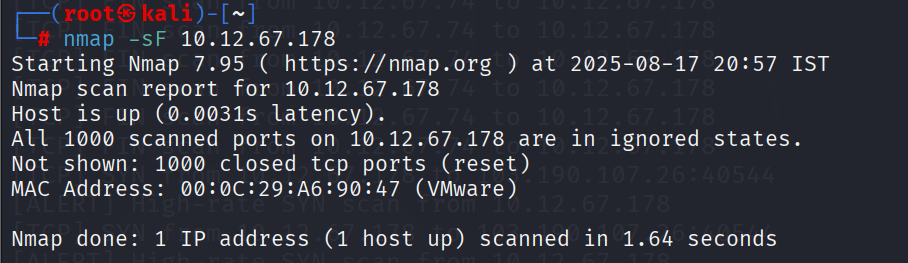
Test 4 – NULL Scan



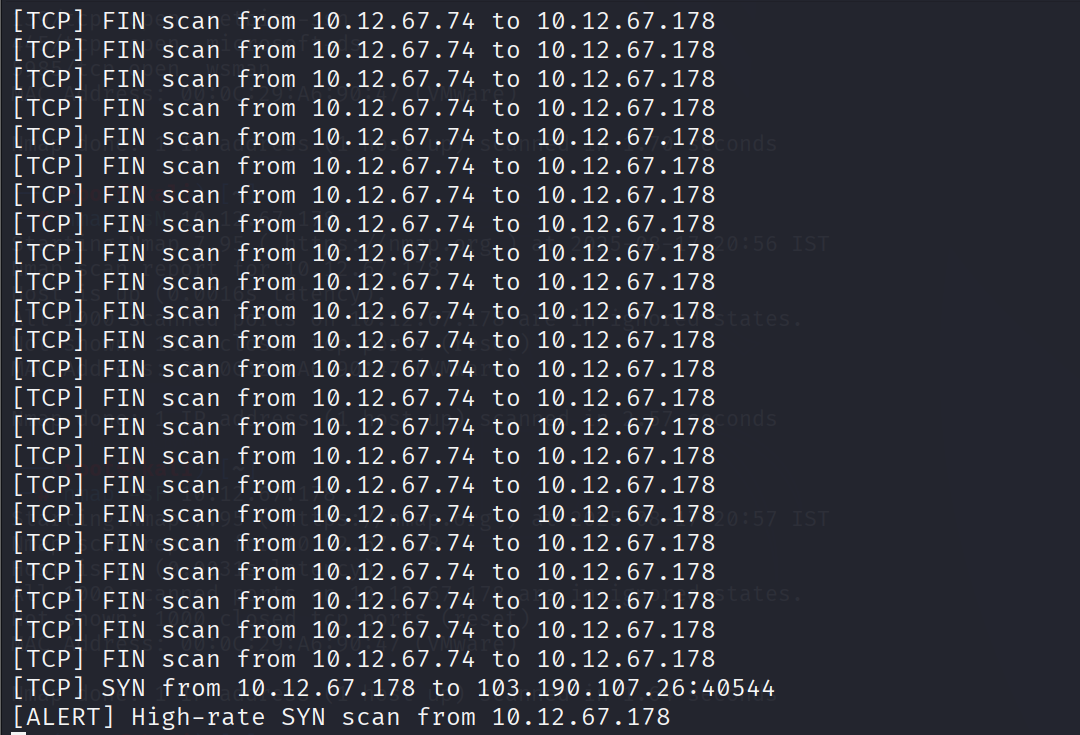
Expected: NIDS logs **NULL scan attempts**.



Test 5 – FIN Scan



Expected: NIDS logs **FIN scan attempts**.



## ****6. Detection & Mitigation Notes****

**False Positives:**

Normal admin tools (ping sweeps, port scans) may trigger alerts.

**Mitigation:**

Add configurable thresholds.

Log alerts into files or a database for SIEM integration.

**Next Steps:**

Implement protocol-specific anomaly detection (DNS tunneling, ARP spoofing).

Add support for alerting via email/Slack.

Extend detection using signatures (Snort-like rules).

## ****7. Conclusion****

This PoC successfully demonstrates the creation of a simple yet effective Network Intrusion Detection System (NIDS). The tool was able to detect ICMP pings, SYN floods, and scanning techniques such as NULL and FIN scans in a lab environment.  
By extending its detection logic and integrating alerting mechanisms, this PoC provides a foundation for developing a more robust intrusion detection solution suitable for enterprise networks.