

PoC - Lightweight Network IDS

Name : Shivani Yadav

Intern ID : 279

Objectives

The aim of this project is to build a **lightweight Network IDS using Python and Scapy** that can identify common suspicious activities from packet capture files. The IDS should:

1. Detect ICMP echo requests and replies (ping traffic)
2. Track TCP connection attempts (SYN packets)
3. Identify scanning behavior (SYN, NULL, and FIN scans)
4. Flag abnormal high-volume activities (possible floods or repeated probes)

1. Tool Overview

Name: Simple Network IDS (Python + Scapy)

Purpose: Monitor recorded network traffic and generate alerts for suspicious patterns.

Input: `.pcap` file containing captured packets.

Output: Alerts shown on the console, displaying source/destination IPs, ports, and type of activity.

2. How the Code Works

Step 1 – Import Libraries

Load Scapy to handle packet analysis:

```
from scapy.all import rdpcap, TCP, ICMP
```

Step 2 – Load Traffic File

Prompt user for a `.pcap` file and read all packets:

```
file = input("Enter PCAP file: ")
packets = rdpcap(file)
```

Step 3 – Analyze Each Packet

- **ICMP:** Check for echo request/reply → log ping activity.
- **TCP SYN:** Log new connection attempts.
- **TCP NULL/FIN:** Mark unusual scanning behavior.

Step 4 – Count and Monitor Activity

Maintain counters for ICMP requests, SYN attempts, and special TCP flags.

Step 5 – Generate Alerts

If thresholds are exceeded (too many SYNs, floods of ICMP, or multiple scans), display alerts in the console.

3. Sample Output

```
Enter PCAP file: fin_scan.pcap
[*] Reading packets from fin_scan.pcap...

[TCP] SYN attempt from 10.10.1.5 to 10.10.1.20:22
[TCP] SYN attempt from 10.10.1.5 to 10.10.1.20:80
[ICMP] Echo request from 192.168.0.2 to 192.168.0.3

[ALERT] FIN scan detected from 10.10.1.5
[ALERT] Possible high-rate SYNs from 10.10.1.5
[+] Analysis finished.
```

4. Giving Input

When executed, the script asks for:

Enter PCAP file:

Type the filename (e.g., `traffic_test.pcap`) and press Enter.

Place the `.pcap` file in the same folder as the Python script to avoid path errors.

5. Example PCAP Files for Testing

File	What It Demonstrates
<code>ping_test.pcap</code>	Simple ICMP ping requests and replies
<code>syn_attempts.pcap</code>	TCP SYN packets to multiple ports
<code>null_scan.pcap</code>	Packets with no TCP flags (NULL scan)
<code>fin_scan.pcap</code>	Packets with only FIN flag set
<code>dos_flood.pcap</code>	Excessive ICMP or SYN packets (flooding)

6. What You Will See

- `[ICMP] Echo request from <src> to <dst>`
 - `[ICMP] Echo reply from <src> to <dst>`
 - `[TCP] SYN attempt from <src> to <dst>:<port>`
 - `[ALERT] NULL/FIN scan pattern from <src>`
 - `[ALERT] Abnormal high-rate SYN/ICMP activity from <src>`
-

7. Running the Script

1. Open IDLE or a terminal window.
 2. Load the script (e.g., `simple_ids.py`).
 3. Run the program.
 4. Enter the PCAP file name when prompted.
 5. Observe alerts printed in the shell/console.
-

8. Screenshots


```
PS C:\Users\3520 i5 16GB> pip install scapy
Collecting scapy
  Downloading scapy-2.6.1-py3-none-any.whl.metadata (5.6 kB)
  Downloading scapy-2.6.1-py3-none-any.whl (2.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 2.2 MB/s eta 0:00:00
Installing collected packages: scapy
Successfully installed scapy-2.6.1



[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\3520 i5 16GB> |
```

```
Enter PCAP file path: nmap_zombie_scan.pcap
[*] Reading packets from nmap_zombie_scan.pcap...
[TCP] SYN attempt from 192.168.100.101 to 192.168.100.102:80
[TCP] SYN attempt from 192.168.100.101 to 192.168.100.102:80

[+] Analysis complete.
```

Source of the .pcap files from Wireshark Nmap-Captures.zip

 [NMap Captures.zip](#) (libpcap) Some captures of various [NMap](#) port scan techniques.

 nmap_OS_scan		15-08-2025 07:24 PM	Wireshark capture ...	158 KB
--	---	---------------------	-----------------------	--------

[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:9594
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:9207
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:10024
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:2557
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:7200
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:2601
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:7004
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:10002
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:787
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:1999
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:10621
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:9071
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:9998
[ALERT] Possible SYN scan from 192.168.100.103
[ALERT] High-rate SYNs from 192.168.100.103
[TCP] SYN attempt from 192.168.100.103 to 192.168.100.102:61532

9. Advantages & Future Improvements

Advantages:

- Very simple and lightweight (runs with just Python + Scapy).
- Easy to demonstrate using small `.pcap` files.
- Provides quick visibility into suspicious network behaviors.
- Good learning project for understanding how IDS works.

Limitations:

- Limited to basic detection; may generate false positives.
- Works only on `.pcap` files (not real-time traffic in this PoC).
- Cannot prevent attacks, only detect them.

Future Improvements:

- Extend to real-time packet sniffing using Scapy's `sniff()` function.
- Export alerts to log files for further analysis.
- Add support for detecting UDP scans and abnormal payload patterns.
- Create a simple GUI or web dashboard to display alerts.

What is the Objective of this PoC?

The purpose of this Proof of Concept is to demonstrate a lightweight IDS built in Python that analyzes packet captures and detects ICMP traffic, TCP connection attempts, and simple port scans. It shows how suspicious network behavior can be identified quickly in a controlled test environment.