Name: Chaitali Patil
Intern ID: 297

Lightweight Network IDS

1.Objective

The primary objective of this Proof of Concept (PoC) is to design and demonstrate a Python-based Lightweight Network Intrusion Detection System (IDS).

The system analyzes network traffic from packet capture (PCAP) files and identifies:
- ICMP traffic: Detecting ping requests/replies used for host discovery.
- TCP SYN packets: Recognizing new connection attempts that could indicate port scanning.
- Stealth scanning techniques: Such as SYN, NULL, and FIN scans.
- Abnormally high traffic rates: Identifying potential Denial-of-Service (DoS) or brute-force scanning attempts.

This PoC shows how malicious or suspicious network behavior can be detected using simple Python logic in a controlled lab environment.

2. Tool Overview

- Tool Name: Python Network IDS (Scapy-based)
- Technology Stack: Python + Scapy library
- Input: .pcap files (captured network traffic using Wireshark/tcpdump)
- Output: Real-time alerts in the console, indicating suspicious activity with source/destination IPs.
- Purpose: Educational demonstration of IDS fundamentals before deploying enterprise-level tools like Snort or Suricata.

3. Methodology (How the IDS Works)

The IDS follows a step-by-step detection process:

1. Packet Reading – The script reads packets from the provided .pcap file using rdpcap() from Scapy.
2. Packet Analysis – Each packet is checked for its protocol and flags:
   - ICMP → Echo request/reply alerts.
   - TCP SYN → Connection attempt alerts.

- TCP with NULL/FIN flags → Suspicious scan alerts.

3. Suspicious Pattern Detection –
   - Repeated TCP SYN packets from the same source IP may indicate a SYN flood.
   - Multiple different ports probed in quick succession indicate port scan detection.
4. Alert Generation – Alerts are printed in the console with relevant details (source, destination, port, and behavior).
5. Rate Tracking – The system maintains counters to identify high-rate traffic, triggering warnings about floods or brute-force scans.

4. Demonstration Flow

Step 1: Run the script in Python IDLE or terminal.
Step 2: Enter the .pcap file name (e.g., nmap_scan_syn.pcap).
Step 3: The IDS processes the file and prints alerts.

Example IDS Output:
[ICMP] Ping request from 192.168.1.10 to 192.168.1.20
[TCP] SYN attempt from 192.168.1.10 to 192.168.1.20:80
[ALERT] Possible port scan detected from 192.168.1.10
[ALERT] High-rate SYN activity detected from 192.168.1.10

5. Sample PCAP Files Used

PCAP File	What It Demonstrates
nmap_scan_ping.pcap	ICMP ping sweep detection
nmap_scan_syn.pcap	SYN scan on multiple ports
nmap_scan_null.pcap	NULL scan with no TCP flags
nmap_scan_fin.pcap	FIN scan attempt
nmap_zombie_scan.pcap	Repeated SYNs (advanced stealth scan)

6. Observations

- ICMP packets were correctly identified as ping requests.
- TCP SYN detection helped recognize port scans.
- NULL and FIN scans were detected as uncommon flags.
- Continuous SYN floods triggered high-rate suspicious activity alerts.

The system was effective in identifying both simple and stealthy attacks.

7. Conclusion

This PoC demonstrates that even a lightweight Python-based IDS can effectively detect unusual and malicious network activities. While suitable for educational and lab purposes, a production-grade IDS (such as Snort or Suricata) would require additional advanced features such as:

- Real-time packet monitoring
- Logging and reporting
- Signature-based and anomaly-based detection
- Integration with SIEM tools