Proof of Concept- Network IDS

Name- Chanchal Teli

Intern ID- 125

1. Executive Summary

This Proof of Concept (PoC) demonstrates a lightweight Network Intrusion Detection System (NIDS) developed in Python. The system is capable of monitoring both live network traffic and packet capture files (PCAPs), and raising alerts for common reconnaissance and denial-of-service techniques such as:

- ICMP echo requests (pings) and ICMP floods
- TCP SYN connection attempts and SYN floods
- SYN scan behavior across multiple ports
- Suspicious half-open TCP connections

The PoC was executed over a one-week duration and successfully detected controlled test scenarios (ping flood, SYN scan, and half-open connections). While it is not a production-ready IDS like Snort or Suricata, it establishes a strong learning platform and a foundation for further development.

2. Objectives

The PoC was undertaken with the following goals:

1. Develop a working IDS prototype using Python and the Scapy library.
2. Detect basic attack behaviors at the network layer.
3. Provide a demonstration using controlled traffic scenarios.
4. Identify potential false positives and propose improvements.
5. Produce a clear report for documentation and future reference.

3. Approach

3.1 Technology & Tools

- Language: Python 3
- Library: Scapy (packet capture and parsing)
- Environment: Linux workstation with root access for sniffing
- Test Tools: Ping, Nmap (for SYN scans), and sample PCAP files

3.2 Detection Logic

- ICMP Floods: More than 20 echo requests in 10 seconds from a single source triggers an alert.
- SYN Floods: More than 30 SYN packets in 10 seconds from a single source raises a high SYN rate alert.
- SYN Scans: If one source probes more than 10 unique ports in 10 seconds, a SYN scan alert is triggered.
- Half-Open Connections: If more than 70% of SYNs remain unacknowledged, a half-open connection alert is raised.

3.3 Architecture

1. Packet Capture Layer → Reads traffic live or from PCAP.
2. Detection Layer → Uses counters and thresholds to identify anomalies.
3. Alerting Layer → Prints timestamped alerts for easy demonstration.

4. Results

Two types of scenarios were tested:

- Normal Traffic: Benign browsing and DNS traffic produced no false alerts.
- Malicious Traffic:
    - Ping flood generated an ICMP Flood Alert.
    - Nmap SYN scan triggered a SYN Scan Alert.
    - Half-open SYN test triggered a High Half-Open Ratio Alert.

The IDS consistently raised alerts for the expected behaviors, demonstrating that the detection logic works as intended.

5. Limitations & Risks

- False Positives:
    - Monitoring systems may appear as ping floods.
    - Load balancers may generate high SYN rates.
    - Security tools (e.g., vulnerability scanners) may resemble port scans.
- Performance:
    - Current prototype is single-threaded.
    - Suitable for small to medium traffic loads only.
- Scope:
    - Only detects a few network-layer techniques.
    - No support for deep packet inspection or application-layer attacks.

6. Recommendations & Next Steps

1. Add support for UDP anomaly detection (amplification attempts).
2. Store alerts in log files or integrate with SIEM systems.
3. Build a simple dashboard for visualization.
4. Introduce whitelisting for trusted hosts to reduce false positives.
5. Explore integration with MITRE ATT&CK tactics for structured mapping.