Proof of Concept — Network IPS

NAME: CHAITANYA MAYEKAR

INTERN ID: 253

# 1. Overview

This Proof of Concept demonstrates a lightweight Intrusion Prevention System (IPS) capable of detecting and blocking common malicious traffic patterns.  
  
The IPS supports:  
- Live blocking on Linux (via iptables + NFQUEUE).  
- Offline analysis of PCAP files for portability and testing (cross-platform).  
- Replay mode for simulating prevention logic without altering system traffic.  
  
The PoC validates the following detection/prevention mechanisms:  
- ICMP ping floods  
- TCP SYN floods and half-open connection abuse  
- Common scan techniques (NULL, FIN, XMAS, repeated port attempts)  
- Naïve SQL injection payloads in HTTP requests

# 2. Environment Setup

## Linux (Full functionality)

1. Clone/unzip the project.  
2. Create a virtual environment:  
 python3 -m venv .venv && source .venv/bin/activate  
3. Install dependencies:  
 pip install -r requirements.txt  
4. Run analysis on a PCAP (no root required):  
 python -m ips.main analyze --pcap examples/normal.pcap  
5. Enable live blocking (requires root):  
 sudo ./scripts/iptables\_enable.sh  
 sudo -E python -m ips.main live --queue 3  
6. Disable live blocking when finished:  
 sudo ./scripts/iptables\_disable.sh

## Windows (Offline analysis & replay only)

Linux-specific components (nfqueue\_runner.py, iptables scripts, netfilterqueue dependency) are removed.  
  
Steps:  
1. Extract the Windows ZIP.  
2. Create & activate virtualenv (python -m venv .venv).  
3. Install requirements.  
4. Place PCAPs under examples\.  
5. Run:  
 python -m ips.main analyze --pcap examples\normal.pcap  
 python -m ips.main analyze --pcap examples\malicious.pcap

# 3. Prevention Logic

ICMP Flood:  
- Tracks per-source ICMP packets per second.  
- Drops traffic exceeding configured threshold.  
  
SYN Flood / Half-Open Tracking:  
- Tracks outstanding SYN packets.  
- Blocks sources with excessive SYNs not followed by ACKs.  
  
Scan Detection:  
- Identifies abnormal TCP flag combinations (NULL, FIN, XMAS scans).  
- Detects repeated port sweep attempts from same source.  
  
HTTP Payload Signatures:  
- Regex matching for SQL injection patterns such as: union select, ' or 1=1.  
- Encoded payloads in HTTP URI/body (future work).

# 4. False Positives / Tuning

Aggressive rate-limits may incorrectly flag high-frequency but legitimate services (e.g., monitoring pings, load balancer health checks).  
  
Thresholds configurable via CLI/env vars.  
  
Planned features: whitelisting trusted sources and configurable rule engine.

# 5. Results (Demo PCAPs)

Normal PCAP:  
- Packets processed: ~N  
- Drops: 0 (all traffic benign).  
  
Malicious PCAP:  
- ICMP flood detected → packets dropped (ruleId=icmpFlood).  
- SYN flood detected → packets dropped (ruleId=synFlood).  
- Scan attempts detected (ruleId=scanPattern).  
- SQLi payload blocked (ruleId=httpSQLi).  
  
Example alert entry (logs/alerts.jsonl):  
{"ts":"2025-08-29T14:32:01Z","src":"192.168.1.50","dst":"192.168.1.10","proto":"TCP","sport":43210,"dport":80,"action":"DROP","ruleId":"httpSQLi","reason":"Suspicious SQLi payload detected"}

# 6. Next Steps

- Improve TCP state tracking with SYN cookie integration.  
- Expand HTTP inspection with URI decoding and method-aware parsing.  
- Support external configuration files for easier tuning.  
- Investigate eBPF/XDP implementation for higher-performance live blocking.

# 7. Windows Demo Notes

Run analyze on provided normal.pcap and malicious.pcap to generate logs/alerts.jsonl.  
Include excerpts or screenshots in reports for demonstration.