Technical Report: High-Performance IPS Deployment using Snort 3

Subject: Network Security Infrastructure

(**Project:** Inline Intrusion Prevention System (IPS

Date: December 25, 2025

Executive Summary .1

This technical report details the architecture, configuration, and validation of an Intrusion Prevention System (IPS) based on Snort 3. The project successfully transitioned from a passive detection environment (IDS) to an active prevention environment (IPS) using a transparent bridge model. Key results include 100% mitigation of automated reconnaissance .and DoS attacks with minimal impact on network throughput

Technical Architecture & Design .2

The Transparent Bridge Model 2.1

.To ensure maximum security, the system was designed as a **Layer 2 Transparent Bridge**

- **Mechanism:** The IPS node links two physical/virtual interfaces (eth0 and eth1) into a .logical bridge
- **Advantage:** Since the bridge does not require an IP address, it is invisible to attackers .(Stealth Mode), making it harder to bypass or target directly

Data Acquisition (DAQ) Layer 2.2

.The core of the prevention capability lies in the **LibDAQ 3.x** framework

- .**Module:** AFPacket
- **Operation:** Unlike PCAP which only copies packets, AFPacket in inline mode allows Snort to intercept the packet buffer. The command -Q (Queue mode) triggers the decision .engine to either forward the packet to the next interface or drop it entirely

Implementation Details .3

(Kernel Optimization (Solving Kernel Leakage 3.1

A common failure in Linux-based IPS is "Kernel Leakage," where the OS forwards packets .before Snort can inspect them

- .**Solution:** - Disabled IPv4 Forwarding: sysctl -w net.ipv4.ip_forward=0

- ○ .Flushed IP tables and interface addresses to force raw frame handling

(Snort 3 Configuration (Lua Based 3.2

.Snort 3 uses Lua for configuration, allowing for modular and faster processing

- **Key Configuration:**

```
daq = {
module_dirs = { '/usr/local/lib/daq' },
module = 'afpacket',
input_spec = 'eth0:eth1',
{ '    variables = { 'replace=1
{
```

(Threat Mitigation Strategy (Rule Engineering .4

We transitioned from alert to drop actions. The rule logic was tuned to prevent false positives :while maintaining strict security

Action	Logic	Threat Type	Rule ID
DROP	Threshold based	ICMP Flood	1000001
DROP	Detects SYN flags to Port 22	Port Scanning	1000007
REJECT	Pattern matching in Payload	Exploit Attempt	1000015

Performance and Validation Results .5

Mitigation Efficiency 5.1

:During testing with hping3 and nmap, the system demonstrated

- **Reconnaissance Defense:** Nmap scans resulted in Filtered state, meaning the IPS dropped the probe packets without sending an ICMP unreachable response (Standard .(Stealth behavior
- **DoS Resilience:** Under an ICMP flood of 10,000 pps, the IPS maintained stability, .dropping 99.8% of malicious traffic while allowing legitimate TCP sessions to persist

Latency Analysis 5.2

Using ping through the bridge, the RTT (Round Trip Time) increased by only **~1.8ms to 2.2ms** compared to a direct link, which is well within acceptable parameters for enterprise-grade security appliances.

6. Challenges and Troubleshooting

1. **Interface Desync:** Initially, Snort would only block one-way traffic.
    - **Fix:** Corrected the DAQ spec to eth0:eth1 (bidirectional) and ensured both interfaces were in promisc (promiscuous) mode.
2. **Resource Contention:** High CPU usage during heavy loads.
    - **Fix:** Optimized Snort 3 thread affinity to pin the process to specific CPU cores.

7. Conclusion

The implementation confirms that Snort 3 is a robust, modern solution for active network defense. By utilizing a transparent bridge and the AFPacket DAQ, we achieved a high-security posture that effectively neutralizes threats at the link layer before they reach the application layer.

8. Appendix: Command Reference

- **Execution:** snort -c /etc/snort/snort.lua -Q --daq afpacket -i eth0:eth1
- **Verification:** tail -f /var/log/snort/alert_fast