Snort IDS Project : 2 Report

Title: Snort IDS – False Positive Optimization

Author: Taranjyot Kaur Kathoda

This project is a continuation of my first Snort IDS project.
In the first project, I implemented multiple Snort rules to detect different types of network attacks. While testing those rules, I noticed that Snort was generating **too many alerts**, even for normal traffic.

So in this project, I focused on **reducing false positives** by optimizing a few rules instead of adding new complex detections.

Project Goal

The main goal of this project is to:

- Understand what false positives are in IDS
- Learn why poorly written rules generate alert noise
- Optimize basic Snort rules so that alerts are generated only when required
- Keep everything at a **beginner-friendly level**

This project does **not** use machine learning or SIEM tools.
It only focuses on **simple rule tuning techniques**.

Lab Setup

- IDS Machine: Kali Linux (Snort installed)
- Attacker Machine: Ubuntu Linux
- Target Machine: Linux Mint
- Virtualization: Oracle VirtualBox
- Tools Used:
    - Snort
    - Nmap
    - Ping

All machines are connected on the same internal network.

What are False Positives?

A false positive happens when normal or harmless network traffic is detected as an attack.

Examples:

- A normal ping generating an alert
- Basic network scanning triggering multiple alerts
- Repeated alerts for the same activity

Too many false positives make IDS alerts difficult to analyze.

Rules Optimized in This Project

In this project, I optimized **two basic Snort rules**.

1. ICMP Reconnaissance Rule

Problem

Initially, Snort was generating an alert for **every single ping request**, even during normal connectivity testing.

Solution

I added a detection filter so that Snort only generates an alert when **multiple ICMP packets** are sent within a short time.

Optimized Rule

```
alert icmp any any -> $HOME_NET any
(
 msg:"ICMP Recon Detected";
 itype:8;
 detection_filter:track by_src, count 5, seconds 20;
 sid:2000001;
 rev:2;
)
```

Now, a single ping does not generate an alert, but continuous pinging is detected as reconnaissance.

2. TCP Port Scan Detection Rule

Problem

During Nmap scans, Snort generated too many alerts because each SYN packet was logged separately.

Solution

I optimized the rule using a detection filter so alerts are generated only when multiple SYN packets are detected from the same source.

Optimized Rule

```
alert tcp any any -> $HOME_NET any
(
 flags:S;
 detection_filter:track by_src, count 10, seconds 5;
 msg:"TCP Port Scan Detected";
 sid:2000002;
 rev:2;
)
```

This reduced alert noise and helped detect actual port scanning behavior.

Testing

- ICMP testing was done using continuous ping from the attacker machine.
- TCP scanning was performed using Nmap SYN scans.
- Snort alerts were monitored in real time on the IDS machine.

Screenshots of both the attack and detection are included in this repository.

Results

- False positives were reduced significantly
- Normal traffic no longer triggered alerts

- Only suspicious behavior generated alerts
- Snort became easier to monitor and analyze

What I Learned

- How false positives affect IDS efficiency
- Basics of Snort rule tuning
- Use of detection filters
- Difference between normal and suspicious network behavior
- Practical IDS testing in a lab environment

Limitations

- This project uses signature-based detection only
- Rules are manually tuned
- No automated alert correlation

Conclusion

This project helped me understand that writing IDS rules is not just about detection, but also about **reducing noise**.
By making small changes to basic Snort rules, false positives can be reduced without making the system complex.

This project is intended for learning and beginner-level practice.