Snort Project 1 – Intrusion Detection Using Custom Rules

1. Project Overview

This project focuses on implementing and analyzing **Snort (Version 3)** as a Network Intrusion Detection System (NIDS) in a Kali Linux environment. The objective is to understand how Snort rules work, how alerts are generated, and how custom rules can be written to detect suspicious or malicious network traffic.

Snort is widely used in real-world security operations for real-time traffic analysis and packet logging. In this project, default and custom rules were explored and tested to observe alert behavior.

---

2. Tools & Environment Used

- Operating System: **Kali Linux**
- IDS Tool: **Snort 3**
- Configuration File: snort.lua
- Rules Directory: /etc/snort/rules/
- Editor Used: nano

---

3. Understanding Snort Rules

Snort rules define patterns that Snort uses to detect network activities. Each rule consists of: - **Action** (alert, log, drop) - **Protocol** (TCP, UDP, ICMP) - **Source & Destination IP/Port** - **Rule Options** (message, SID, content, etc.)

Example rule structure:

```
alert icmp any any -> any any (msg:"ICMP Packet Detected"; sid:1000001;)
```

---

4. Rules Implemented in This Project

The following custom rules were implemented as part of this project. All rules were maintained inside the local.rules file for safe customization.

4.1 ICMP (Ping) Detection Rule

alert icmp any any -> 192.168.1.14 any (msg:"Ping is detected"; sid:10000; rev:1;)

Purpose: Detects ICMP ping requests to a specific internal host.

4.2 HTTP/HTTPS Traffic Detection Rule

alert tcp any any -> 10.0.2.15 [80,443] (msg:"HTTP Traffic detected"; sid:100002; rev:1;)

Purpose: Monitors incoming HTTP and HTTPS traffic to the target machine.

4.3 DNS Query Detection Rule

alert udp any any -> any 53 (msg:"DNS Query Detected"; sid:1000003; rev:1;)

Purpose: Detects DNS queries over UDP port 53.

4.4 SSH Brute Force Attempt Detection

alert tcp any any -> any 22 (msg:"SSH Brute Force Attempt"; flow:to_server, established;)

Purpose: Identifies suspicious SSH access attempts that may indicate brute-force behavior.

4.5 TELNET Access Detection Rule

alert tcp any any -> any 23 (msg:"TELNET Attempt Detected"; sid:2000003; rev:1;)

Purpose: Detects TELNET connection attempts, which are considered insecure.

4.6 FTP Traffic Blocking Rule

drop tcp any any -> 10.0.2.15 21 (msg:"Block FTP Traffic"; sid:1000005; rev:1;)

Purpose: Blocks FTP traffic to prevent insecure file transfers.

4.7 SMTP Traffic Rejection Rule

reject tcp any any -> 10.0.2.15 21 (msg:"SMTP Traffic Rejected"; sid:1000006; rev:1;)

Purpose: Rejects unauthorized SMTP traffic to reduce spam or misuse.

4.8 Custom Port Blocking Rule (Port 9999)

block tcp any any -> 10.0.2.15 9999 (msg:"Blocking traffic to port 9999"; sid:1000011; rev:1;)

Purpose: Blocks all traffic destined to a non-standard port for security hardening.

Custom rules were added in local.rules to avoid modifying default rule files.

---

5. Configuration & Execution

Snort was executed using the following command:

sudo snort -c /etc/snort/snort.lua -i <interface> -A alert_fast

Configuration validation was performed using:

sudo snort -c /etc/snort/snort.lua -T

---

6. Alert Generation & Logs

Alerts generated by Snort were stored in:

/var/log/snort/

These alerts confirm that Snort successfully detected network traffic based on the configured rules.

---

7. Learning Outcomes

- Practical understanding of Snort 3 architecture
- Ability to write and manage custom Snort rules

- Hands-on experience with IDS alert generation
- Understanding real-time network traffic monitoring

---

8. Conclusion

This project demonstrates the effective use of Snort as a Network Intrusion Detection System. By creating and testing custom rules, the project highlights how rule-based detection works in real-world cybersecurity environments. This serves as a strong foundation for advanced IDS/IPS and SOC-related roles.
