# Snort Detection Rules Project

## 1. Introduction

- Project Title: Implementing and Testing Detection Rules with Snort
- Objective: I am using Snort because it's a powerful NIDR/NIPS , the purpose of the project is to learn how the rules works and trigger. In this project I will firstly create an alert for DoS attacks and then I will make a rule to block DoS attacks
- Scope: Detect DoS attack

A Denial-of-Service (DoS) attack is when an attacker floods a target system or network with excessive requests to exhaust its resources, making it unavailable to legitimate users.

## 2. Background & Motivation

- Snort is a Network Intrusion Detection System (NIDS) that monitors network traffic between endpoints and the internet. It detects anomalies and potential threats based on predefined or custom detection rules. Snort provides an additional layer of security by identifying suspicious activities in real-time. If an incident occurs, Snort logs alerts, which can be used to analyze and correlate events based on timestamps and triggered rules, helping security teams understand what actions the attacker attempted on the network. Detection Rules in Snort: Explain how detection rules work and why they are critical for intrusion detection systems (IDS).
- Why this Project? This project aims to explore Snort's capabilities as a NIDS by creating and testing custom detection rules. Through this, I will demonstrate my understanding of network security, Linux administration, and intrusion detection

techniques, while gaining practical skills in analyzing and mitigating cyber threats."

### 3. Environment Setup

- System Requirements
  - Kali linux VM1, Xubuntu VM2
- Tools and Dependencies
  - Snort version
  - Supporting tools tcpdump for trouble shooting

### 4. Creating Snort Rules

Snort rules are made by 2 main components

| Rule header | Rule options |
|---|---|

**Rule header** has these parameters

| Action | Protocol | Source ip and port | direction | Destination ip and port |
|---|---|---|---|---|

**alert icmp any any -> $HOME_NET any (msg:" ICMP Packet found"; sid:1000001; rev:1;)**

Has you can see in the example we put in action section alert but it can be other options example log, drop, reject.

**alert** will notify us when the rule is triggered and log the event.

**log** will only register the event in a log without notifying us.

**drop** will block the packet from reaching its destination without notifying the sender.

**reject** will block the packet and notify the sender

In protocol we choose the protocol which trigger the rule, it can be icmp , ip , tcp, udp

**ICMP** Used for **network diagnostics**

**TCP** A reliable, connection-oriented protocol that ensures data arrives correctly.

**UDP** A fast, connectionless protocol used for video streaming, VoIP, and gaming.

**IP** It allows Snort to inspect and filter traffic **at the IP level**, regardless of whether the packet contains **TCP, UDP, or ICMP**.

The destination IP and port follow the same logic as the source IP and port. They specify a precise IP and port; otherwise, we can use 'any' to apply the rule to all traffic.

Direction operator indicates the traffic flow to be filtered by Snort.

**->** Source to destination flow.

**<>** Bidirectional flow

**Rules options** can be categorized in different ways, including more complex configurations. However, for the sake of this documentation, we will focus on creating a simple rule.

```
( msg: “ICMP Packet found”; reference:CVE-XXXX; sid:1000001; rev:1; )
```

**msg** The message we receive when the alert is triggered.

**reference** Additional information for the alert. It can be a CVE, an exploit, or a MITRE ATT&CK technique.

**sid** A unique Snort rule ID that helps identify the rule.

**rev** The revision number of the rule. Each time we update or improve the rule, we increase this value.

## 5. Attack Simulations

In my Xubuntu VM

I did sudo nano /etc/snort/rules/local.rules to edit local.rules

```
# additions here.
alert icmp any any -> any any (msg:"ICMP Packet Detected! "; sid:1000001; rev:1;)
```

This is a simple rule to see if our Snort configuration works

Then I use this command to start Snort in IDS mode:

sudo snort -A console -i enp0s3 -c /etc/snort/snort.conf

In my kali VM i did ping -c 5 192.168.1.31 to send icmp signal

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ ping -c 5 192.168.1.31

PING 192.168.1.31 (192.168.1.31) 56(84) bytes of data.
64 bytes from 192.168.1.31: icmp_seq=1 ttl=64 time=1.15 ms
64 bytes from 192.168.1.31: icmp_seq=2 ttl=64 time=0.822 ms
64 bytes from 192.168.1.31: icmp_seq=3 ttl=64 time=0.630 ms
64 bytes from 192.168.1.31: icmp_seq=4 ttl=64 time=0.620 ms
64 bytes from 192.168.1.31: icmp_seq=5 ttl=64 time=0.568 ms

── 192.168.1.31 ping statistics ──
5 packets transmitted, 5 received, 0% packet loss, time 4058ms
rtt min/avg/max/mdev = 0.568/0.757/1.149/0.213 ms

┌──(kali㉿kali)-[~]
```

As you can see my Xubuntu machine has trigger the Alert in Snort

```
        Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
Commencing packet processing (pid=1777)
03/18-13:33:56.569810  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {IPV6-ICMP} fe80::28e6:373b:7d18:6dc7 -> ff02::2
03/18-13:36:29.367554  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [Classification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.250:1900
03/18-13:36:29.367555  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [Classification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.250:1900
03/18-13:36:29.367556  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [Classification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.250:1900
03/18-13:36:29.370571  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [Classification: Misc Attack] [Priority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.250:1900
03/18-13:36:59.619871  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {IPV6-ICMP} fe80::a00:27ff:fe33:657 -> ff02::2
03/18-13:37:02.396713  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.32 -> 192.168.1.31
03/18-13:37:02.396728  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.31 -> 192.168.1.32
03/18-13:37:03.398469  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.32 -> 192.168.1.31
03/18-13:37:03.398498  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.31 -> 192.168.1.32
03/18-13:37:04.399901  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.32 -> 192.168.1.31
03/18-13:37:04.399920  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.31 -> 192.168.1.32
03/18-13:37:05.428291  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.32 -> 192.168.1.31
03/18-13:37:05.428310  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.31 -> 192.168.1.32
03/18-13:37:06.451921  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.32 -> 192.168.1.31
03/18-13:37:06.451940  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {ICMP} 192.168.1.31 -> 192.168.1.32
03/18-13:38:05.842631  [**] [1:1000001:1]  ICMP Packet Detected!  [**] [Priority: 0] {IPV6-ICMP} fe80::28e6:373b:7d18:6dc7 -> ff02::2
^C
```

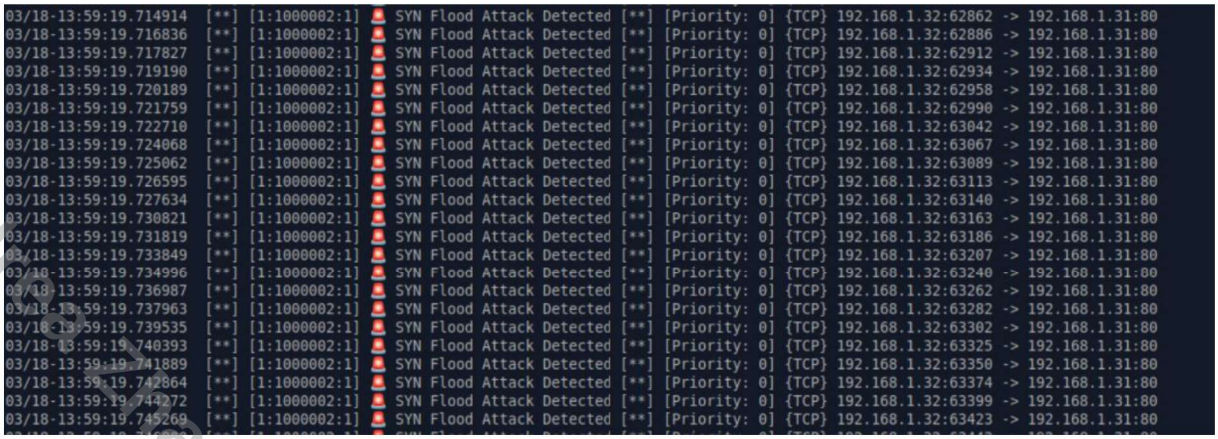
Now let's make a scenario we received a DoS attack

We make a DoS alert rule

```
alert tcp any any -> $HOME_NET any (msg:"🚨 SYN Flood Attack Detected"; flags:S; threshold:type threshold, track by_src, count 20, seconds 3; sid:1000002; rev:1;)
```

Using kali VM this command we simulate a DoS attack

sudo hping3 -S -p 80 --flood 192.168.1.31




We want to stop the DoS attack from that source so I change local.rules

```
drop tcp 192.168.1.32 any -> $HOME_NET any (msg:" SYN Flood Attack from KALI - Dropped";  sid:1000004; rev:1;)
```

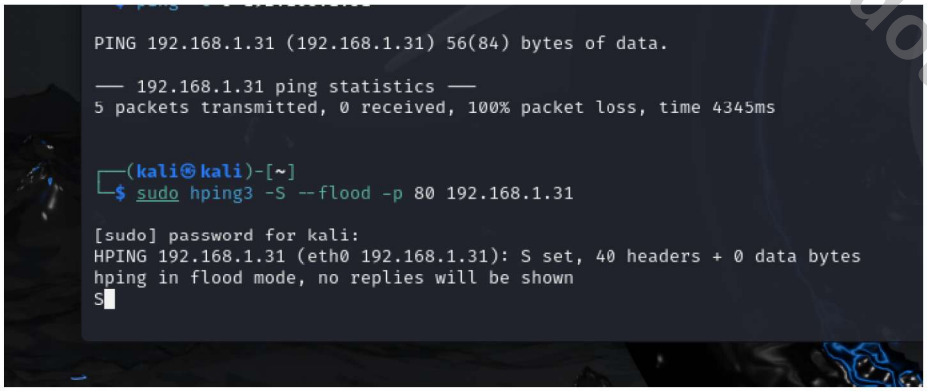
With this rule I will block DoS attempt from my kali VM

Now I run Snort has IPS

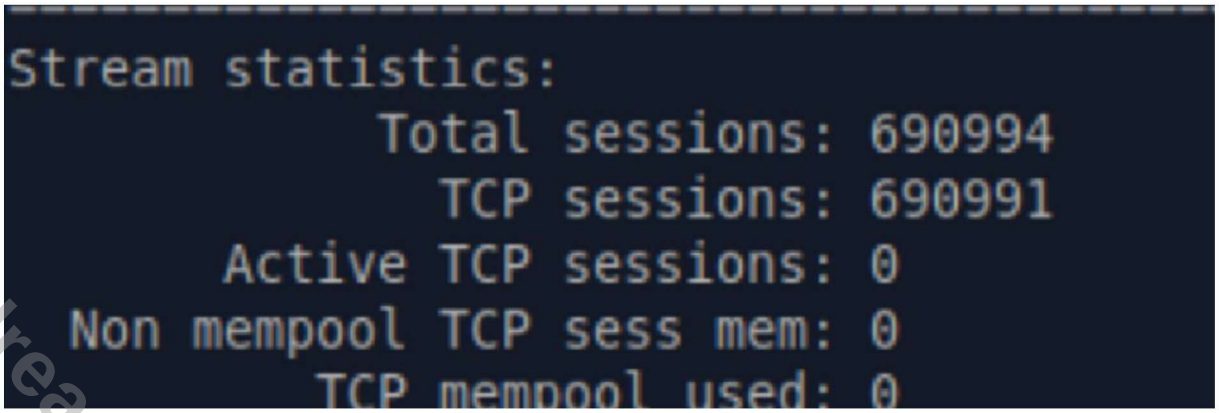
sudo snort -Q --daq nfq --daq-mode inline -c /etc/snort/snort.conf

i sent TCP flood from my kali machine with this command

sudo hping3 -S --flood -p 80 192.168.1.31




Then I stopped IPS and check




The IPS has blocked 690991 TCP flood

## 6. Challenges & Lessons Learned

### Common Issues Faced

Running two virtual machine have to know how to configure the network setup otherwise they cannot communicate.

I also learned how to trouble shoot Snort installation issues and configuration issues.

### Solutions & Best Practices

This was a basic rule but we can make it more improved depends on the organization needs.

## 7. Conclusions

Through this hands-on experience, we successfully configured, tested, and troubleshooted Snort in both IDS and IPS modes, demonstrating deep knowledge of Linux networking, intrusion detection, and prevention systems.

We explored packet analysis, rule creation, attack detection (ICMP, SYN flood, Nmap scans), and IPS enforcement using Snort's alerting and blocking capabilities. Key challenges included DAQ configuration,

iptables redirection, and log monitoring, all of which were resolved through systematic debugging.

This experience showcased proficiency in Linux system administration, network security, and practical threat mitigation, reinforcing our ability to implement and fine-tune IDS/IPS solutions in real-world environments.