Snort: Intrusion Detection and Prevention System

Wissal BOUTAYEB
Fatima BOUYARMANE

Introduction:

In the age of the internet, organizations are heavily relying on IT infrastructure to keep them safe from cyberattacks. As more and more organizations are adopting digital transformation, the risk of cybercrime is increasing at a rapid rate; so is the importance of cybersecurity.

Cybersecurity has become the knight in shining armor. Strong cybersecurity policy and infrastructure work together to secure computer systems and networks from unauthorized access and attacks. Businesses, individuals, and governments are investing heavily to reap the benefits of cybersecurity in protecting their assets and data against hackers. For any business to survive in today's competitive world, it requires the right tools and a cybersecurity strategy.

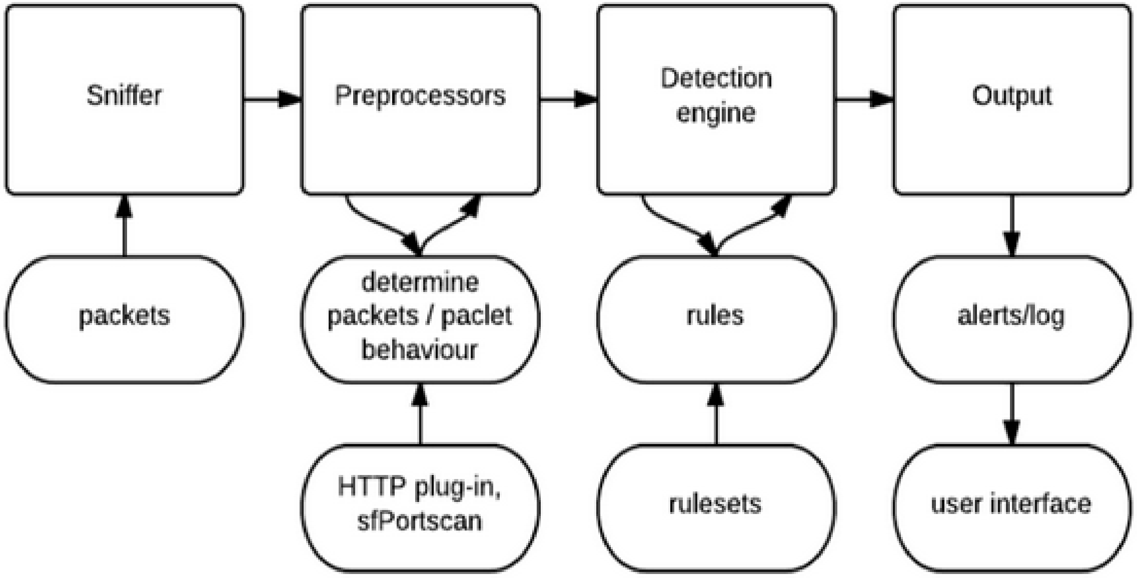
One such indispensable tool in the cybersecurity arsenal is **Snort**. Developed as an open-source intrusion detection and prevention system, Snort plays a crucial role in fortifying network defenses. Its robust architecture and flexible rule sets empower organizations to detect and respond to a wide array of cyber threats effectively.

Objectives:

The primary objectives of our project are to systematically assess the effectiveness of Snort, an Intrusion Detection System (IDS), in responding to diverse cyberattacks. Through the simulation of various attack scenarios, we aim to evaluate Snort's detection accuracy, response time, and adaptability to different threat vectors. The project will involve the creation and refinement of custom Snort rules tailored to the organization's threat landscape, with the overarching goal of providing a comprehensive evaluation report that includes recommendations for optimizing Snort's configurations and rule sets to enhance its overall efficacy in detecting and mitigating cyber threats.

Snort Architecture:

Snort comprises of mainly five components working together to monitor and analyze all network traffic and look for any signs of intrusions and generate alert. As shown in the figure below.



Packet Decoder*:* Captures packets from the network traffic and sets them up for preprocessor or for the detection engine.

Preprocessors**:** Processes the captured packets against certain plugins. These plugins check for known type behavior or anomalies. Preprocessors are indispensable part of any IDS to prepare data packets to be checked by detection engine against rules in the detection engine as intruders may modify those packets to escape any detection.

Detection Engine: Once the data is handled by the preprocessors, it is then passed on to detection engine. The detection engine is the most critical component of the signature-based IDS in Snort. It matches the data packets with the set of rules for any intrusion signatures contained in the data packets. If the rules match the data packets then it is passed on to the alert processor. It may take different amount of time for responding to several types of packets irrespective of the computing system it is running on.

***Logging and Alerting System*:** Generation of alerts and logging is handled by this system. All the alerts and logs are kept in simple plain text files or tcp-dump style files.

Output Module: Output module helps save the logs generated by logging and alerting system in diverse ways like in simple plain text log files, logging to database like MySQL or Oracle or generating XML depending on the configuration set into Snort configuration file.

Key Features:

Packet Sniffing: Snort captures and analyzes network packets in real-time.

Rule-Based Detection: Snort uses rules to identify and alert on suspicious traffic patterns or known attack signatures.

Protocol Analysis: It can perform in-depth analysis of various network protocols to identify anomalies.

Logging and Reporting: Snort generates logs and reports detailing identified threats and network activities.

Types of rules:

Rule Header:

- The rule header contains metadata and defines the rule's action.
- Example: alert ip any any -> any any (msg:"Example Rule"; sid:1000001;)

Rule Options:

- ✓ •alert: Specifies the action to be taken when the rule is triggered (other options include log, pass, activate, dynamic, and drop).
- ✓ ip: Specifies the network layer protocol (other options include tcp, udp, and icmp).
- ✓ any any -> any any: Defines the source and destination addresses and ports.
- ✓ msg:"Example Rule": Describes the purpose or nature of the rule.
- ✓ sid:1000001: Unique identifier for the rule.

Rule options define the conditions that trigger the rule.

- Example: (content:"malware"; nocase; classtype:trojan-activity;)
  - ✓ content:"malware": Specifies the pattern to match in the payload.
  - ✓ nocase: Makes the pattern matching case-insensitive.
  - ✓ classtype:trojan-activity: Classifies the type of activity associated with the rule.

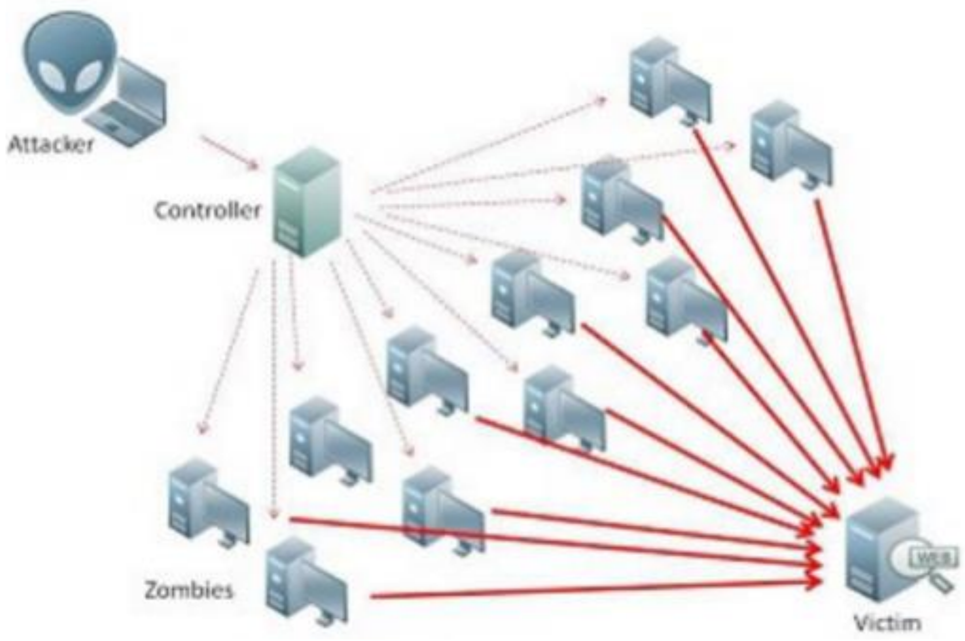
Detecting DoS Attack using Snort:

Abstract - A DoS(denial-of-service) attack is very common and easy to execute and does not require any sophisticated tools. It can happen to anyone. In this project we deploy snort in our home network as a NIDS(Network Intrusion Detection System) to detect a DoS attack and prevent it.

A **denial-of-service attack (DoS attack)** is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the internet. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled.

Denial-of-service attacks are characterized by an explicit attempt by attackers to prevent legitimate use of a service. There are two general forms of DoS attacks: those that crash services and those that flood services. The most serious attacks are distributed.

A **distributed denial-of-service (DDoS)** is a DoS attack where the perpetrator uses more than one unique IP address, often thousands of them. Since the incoming traffic flooding the victim originates from many different sources, it is impossible to stop the attack simply by using ingress filtering. It also makes it very difficult to distinguish legitimate user traffic from attack traffic when spread across so many points of origin. As an alternative or augmentation of a DDoS, attacks may involve forging of IP sender addresses (IP address spoofing) further complicating identifying and defeating the attack. The scale of DDoS attacks has continued to rise over recent years, by 2016 exceeding a terabit per second.



Installation and Configuration:

Installation:

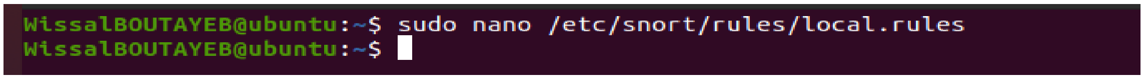
1- **Install Dependencies:**
   - Before installing Snort, ensure that the necessary dependencies are installed. Common dependencies include libpcap, libdnet, and libpcre.

2- **Download Snort:**
   - Download the latest version of Snort from the official website or use a package manager if available for the operating system.

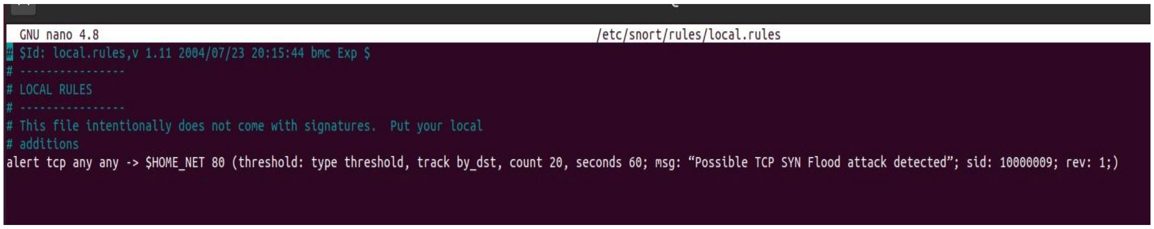
3- **Compile and Install:**
   - Navigate to the Snort directory and follow the installation instructions provided in the documentation. Typically, this involves running ./configure, make, and make install.

Configuration:

- **TCP SYN attack:**



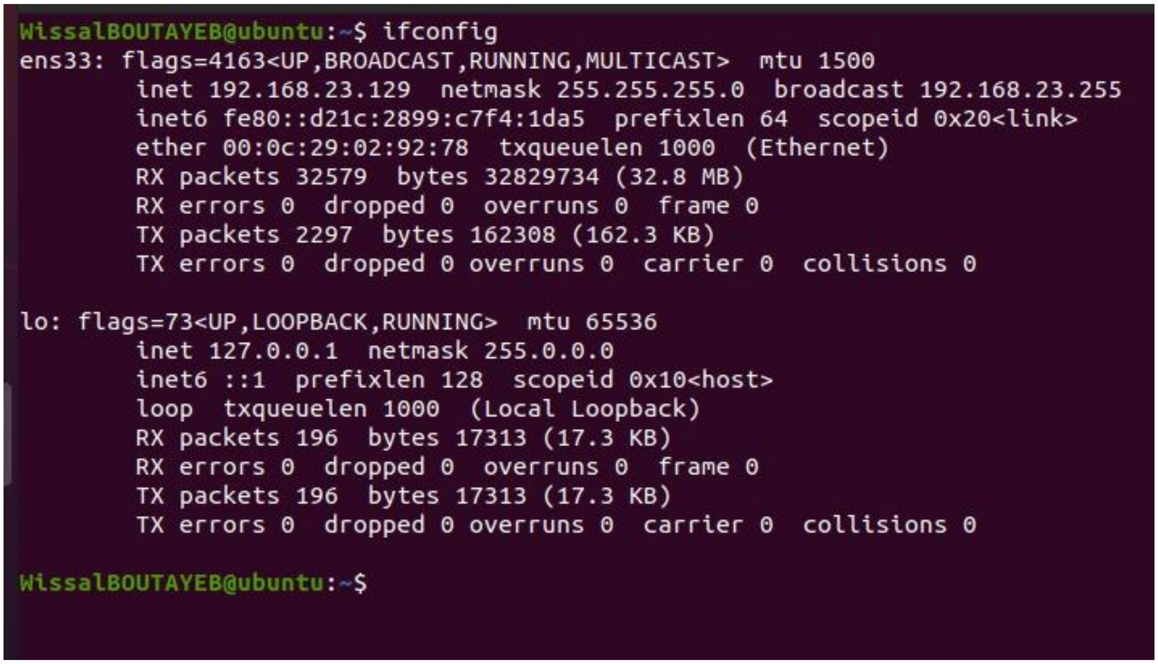
```
WissalBOUTAYEB@ubuntu:~$ sudo nano /etc/snort/rules/local.rules
WissalBOUTAYEB@ubuntu:~$ 
```



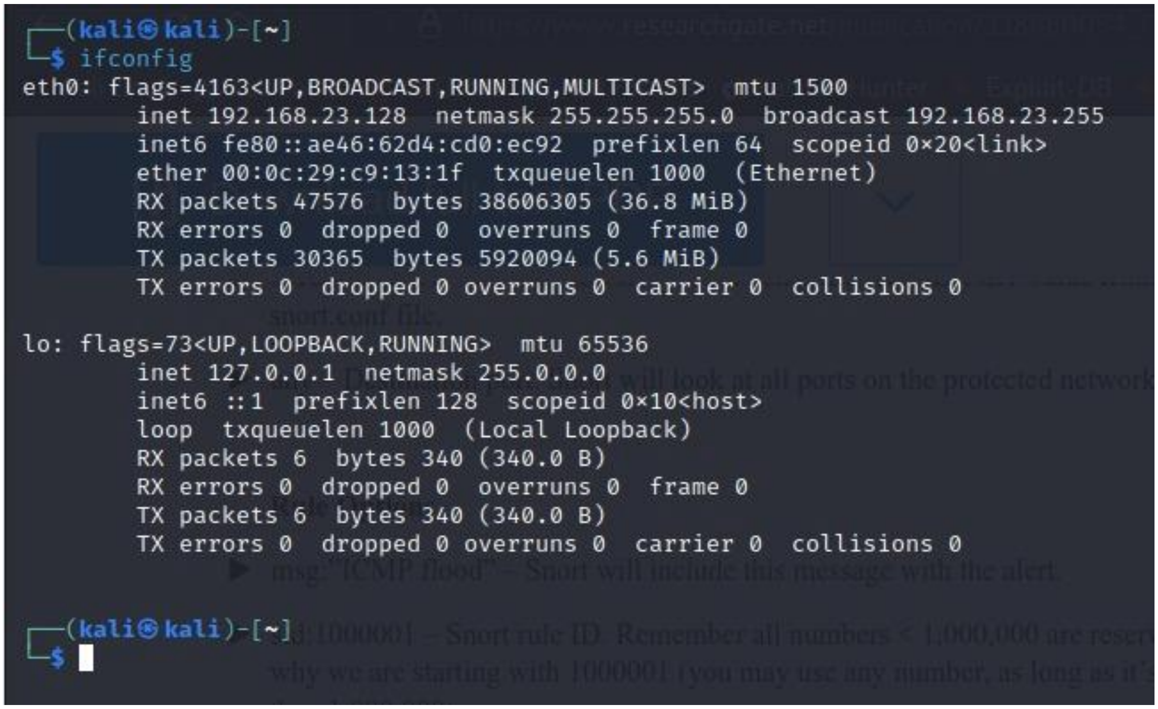
```
GNU nano 4.8                                   /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions
alert tcp any any -> $HOME_NET 80 (threshold: type threshold, track by_dst, count 20, seconds 60; msg: “Possible TCP SYN Flood attack detected”; sid: 10000009; rev: 1;)
```

/etc/snort/rules/local.rules: This is the path to the **local.rules** file. The **/etc/snort/rules/** directory is where Snort typically stores its rule files, and **local.rules** is a file where we can add our custom rules. Therefore, in the picture above we add the rules of **SYN Flood Attack.**

Here you can see that we execute the command ‘ifconfig’ to see the IP addresses of both the Target Computer(Ubuntu) and The computer where we will perform the attack (Kali).

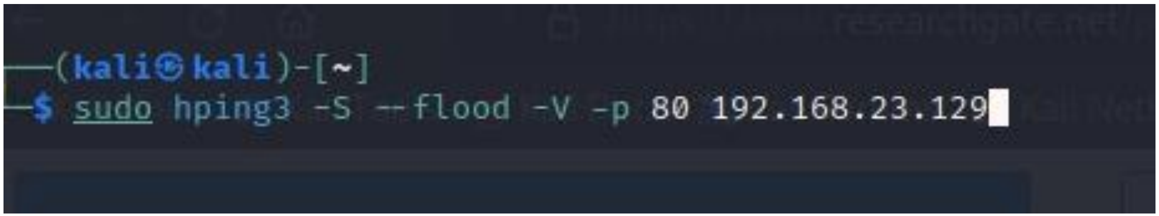


```
WissalBOUTAYEB@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.23.129  netmask 255.255.255.0  broadcast 192.168.23.255
        inet6 fe80::d21c:2899:c7f4:1da5  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:02:92:78  txqueuelen 1000  (Ethernet)
        RX packets 32579  bytes 32829734 (32.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2297  bytes 162308 (162.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 196  bytes 17313 (17.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 196  bytes 17313 (17.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

WissalBOUTAYEB@ubuntu:~$
```

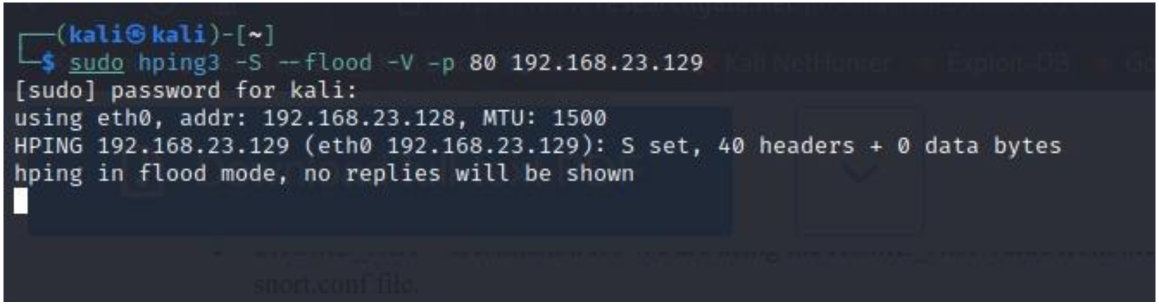


```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.23.128  netmask 255.255.255.0  broadcast 192.168.23.255
        inet6 fe80::ae46:62d4:cd0:ec92  prefixlen 64  scopeid 0×20<link>
        ether 00:0c:29:c9:13:1f  txqueuelen 1000  (Ethernet)
        RX packets 47576  bytes 38606305 (36.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 30365  bytes 5920094 (5.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 6  bytes 340 (340.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 340 (340.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(kali㉿kali)-[~]
└─$ 
```

The command below is using the ‘hping3’ tool to send a flood of TCP SYN packets to the target IP (192.168.23.129)address and port 80 from the attacker computer(kali)

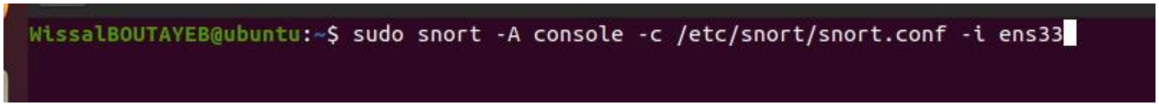


```
──(kali㉿kali)-[~]
─$ sudo hping3 -S --flood -V -p 80 192.168.23.129
```

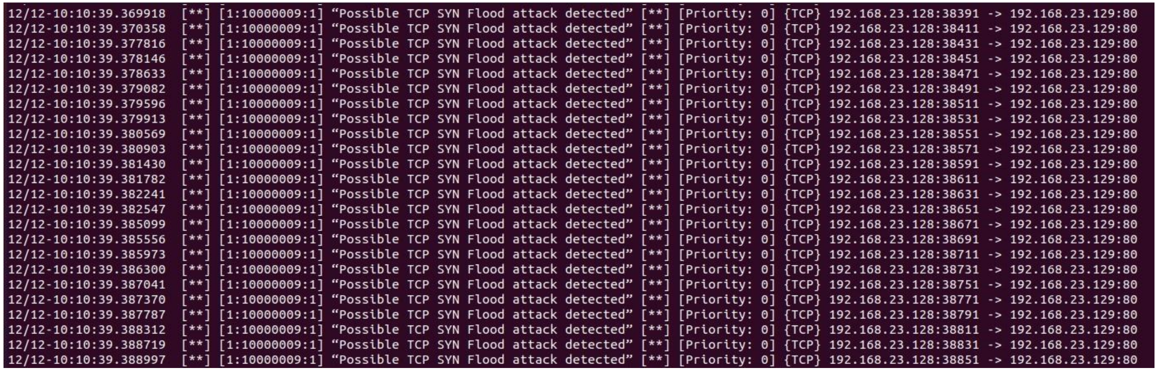


```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -S --flood -V -p 80 192.168.23.129
[sudo] password for kali:
using eth0, addr: 192.168.23.128, MTU: 1500
HPING 192.168.23.129 (eth0 192.168.23.129): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

After performing the attack we execute this command to see the reaction of our intrusion detection system “Snort” by providing the principal file of configuration which is snort. conf and the name of network card in our case is **ens33.**



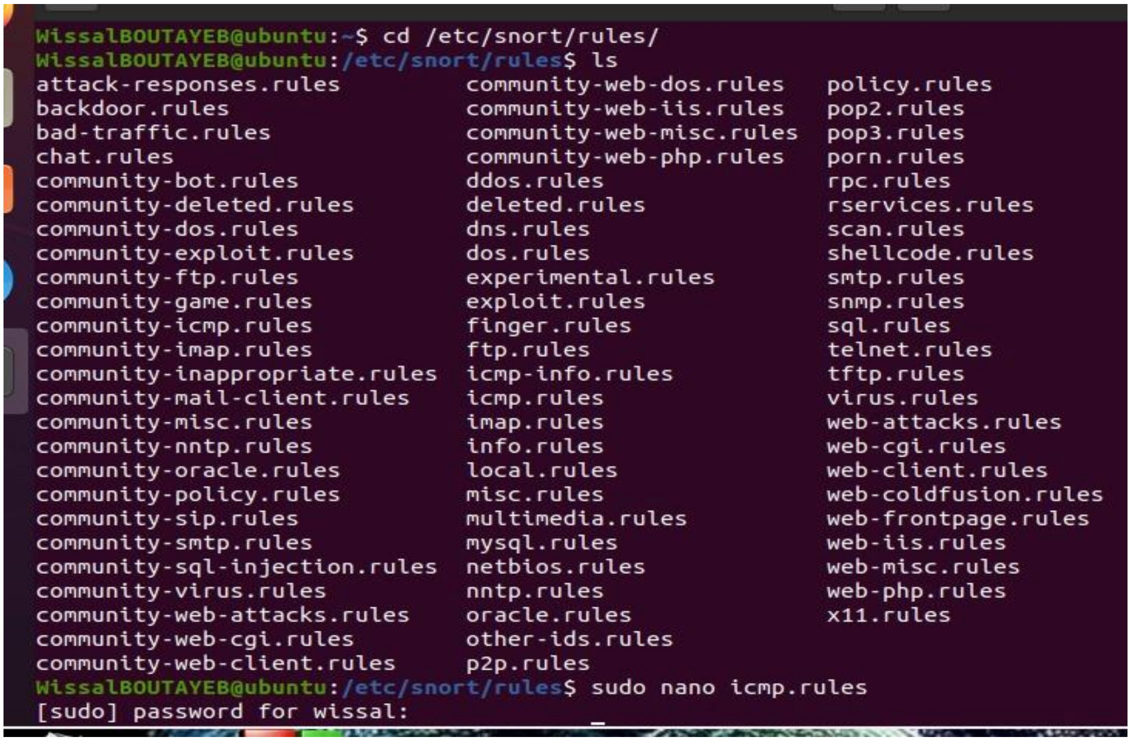
```
WissalBOUTAYEB@ubuntu:~$ sudo snort -A console -c /etc/snort/snort.conf -i ens33
```



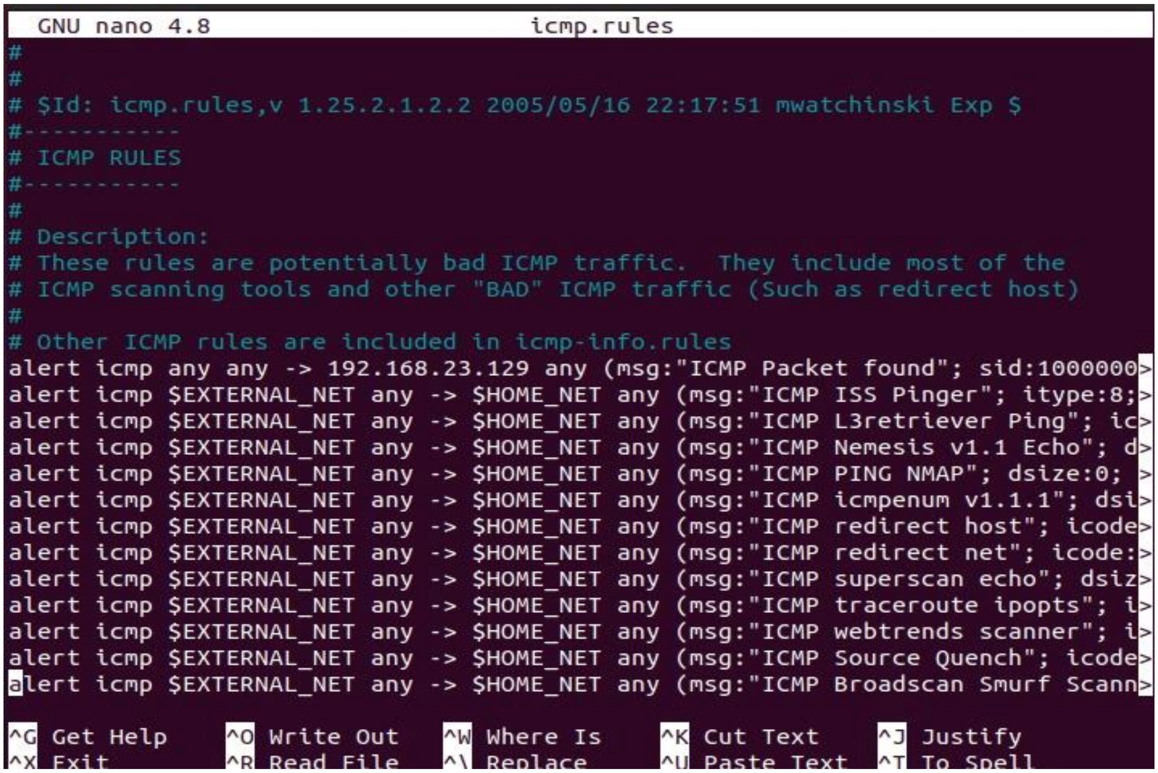
```
12/12-10:10:39.369918  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38391 -> 192.168.23.129:80
12/12-10:10:39.370358  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38411 -> 192.168.23.129:80
12/12-10:10:39.377816  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38431 -> 192.168.23.129:80
12/12-10:10:39.378146  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38451 -> 192.168.23.129:80
12/12-10:10:39.378633  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38471 -> 192.168.23.129:80
12/12-10:10:39.379082  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38491 -> 192.168.23.129:80
12/12-10:10:39.379596  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38511 -> 192.168.23.129:80
12/12-10:10:39.379913  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38531 -> 192.168.23.129:80
12/12-10:10:39.380569  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38551 -> 192.168.23.129:80
12/12-10:10:39.380903  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38571 -> 192.168.23.129:80
12/12-10:10:39.381430  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38591 -> 192.168.23.129:80
12/12-10:10:39.381782  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38611 -> 192.168.23.129:80
12/12-10:10:39.382241  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38631 -> 192.168.23.129:80
12/12-10:10:39.382547  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38651 -> 192.168.23.129:80
12/12-10:10:39.385099  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38671 -> 192.168.23.129:80
12/12-10:10:39.385556  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38691 -> 192.168.23.129:80
12/12-10:10:39.385973  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38711 -> 192.168.23.129:80
12/12-10:10:39.386300  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38731 -> 192.168.23.129:80
12/12-10:10:39.387041  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38751 -> 192.168.23.129:80
12/12-10:10:39.387370  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38771 -> 192.168.23.129:80
12/12-10:10:39.387787  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38791 -> 192.168.23.129:80
12/12-10:10:39.388312  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38811 -> 192.168.23.129:80
12/12-10:10:39.388719  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38831 -> 192.168.23.129:80
12/12-10:10:39.388997  [**] [1:10000009:1] “Possible TCP SYN Flood attack detected” [**] [Priority: 0] {TCP} 192.168.23.128:38851 -> 192.168.23.129:80
```

- ❖ **ICMP Attack:**

For the ICMP attack, we add the ruleset in the file icmp.rules that we find it in /etc/snort/rules



```
WissalBOUTAYEB@ubuntu:~$ cd /etc/snort/rules/
WissalBOUTAYEB@ubuntu:/etc/snort/rules$ ls
attack-responses.rules         community-web-dos.rules   policy.rules
backdoor.rules                 community-web-iis.rules   pop2.rules
bad-traffic.rules              community-web-misc.rules  pop3.rules
chat.rules                     community-web-php.rules   porn.rules
community-bot.rules            ddos.rules                rpc.rules
community-deleted.rules        deleted.rules             rservices.rules
community-dos.rules            dns.rules                 scan.rules
community-exploit.rules        dos.rules                 shellcode.rules
community-ftp.rules            experimental.rules        smtp.rules
community-game.rules           exploit.rules             snmp.rules
community-icmp.rules           finger.rules              sql.rules
community-imap.rules           ftp.rules                 telnet.rules
community-inappropriate.rules  icmp-info.rules           tftp.rules
community-mail-client.rules    icmp.rules                virus.rules
community-misc.rules           imap.rules                web-attacks.rules
community-nntp.rules           info.rules                web-cgi.rules
community-oracle.rules         local.rules               web-client.rules
community-policy.rules         misc.rules                web-coldfusion.rules
community-sip.rules            multimedia.rules          web-frontpage.rules
community-smtp.rules           mysql.rules               web-iis.rules
community-sql-injection.rules  netbios.rules             web-misc.rules
community-virus.rules          nntp.rules                web-php.rules
community-web-attacks.rules    oracle.rules              x11.rules
community-web-cgi.rules        other-ids.rules
community-web-client.rules     p2p.rules
WissalBOUTAYEB@ubuntu:/etc/snort/rules$ sudo nano icmp.rules
[sudo] password for wissal: _
```

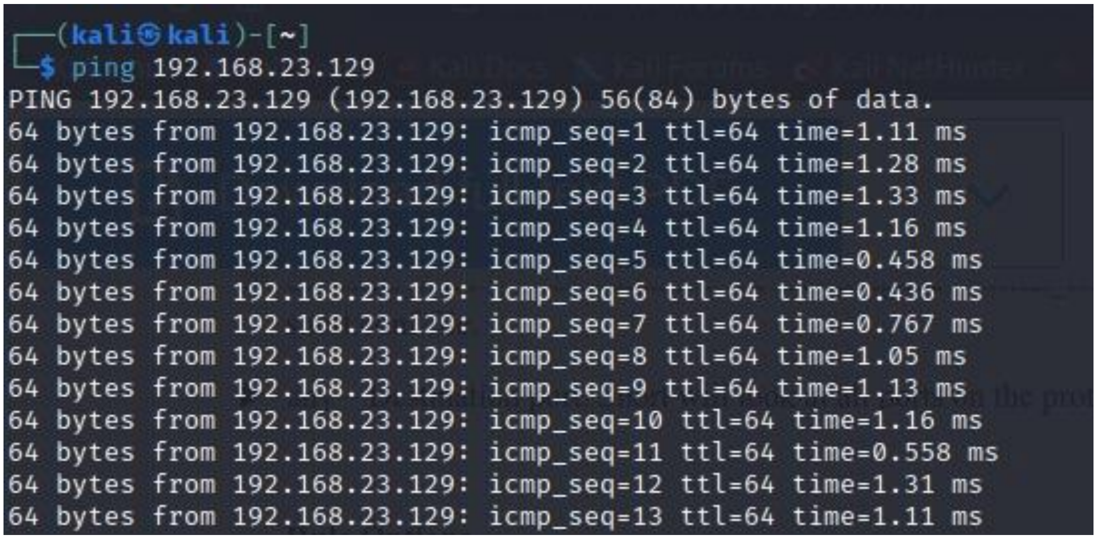


```
  GNU nano 4.8                        icmp.rules
#
#
# $Id: icmp.rules,v 1.25.2.1.2.2 2005/05/16 22:17:51 mwatchinski Exp $
#-----------
# ICMP RULES
#-----------
#
# Description:
# These rules are potentially bad ICMP traffic.  They include most of the
# ICMP scanning tools and other "BAD" ICMP traffic (Such as redirect host)
#
# Other ICMP rules are included in icmp-info.rules
alert icmp any any -> 192.168.23.129 any (msg:"ICMP Packet found"; sid:1000000>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP ISS Pinger"; itype:8;>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever Ping"; ic>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Nemesis v1.1 Echo"; d>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP"; dsize:0; >
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP icmpenum v1.1.1"; dsi>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP redirect host"; icode>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP redirect net"; icode:>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP superscan echo"; dsiz>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP traceroute ipopts"; i>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP webtrends scanner"; i>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Source Quench"; icode>
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Broadscan Smurf Scann>

^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell
```

To detect ICMP echo requests (ping), we use the rule :

Alert icmp any any -> 192.168.23.129 any (msg:"ICMP Packet found"; sid:10000001;)

After adding the rule we perform the attack from kali using the command **ping** and the IP address of the target

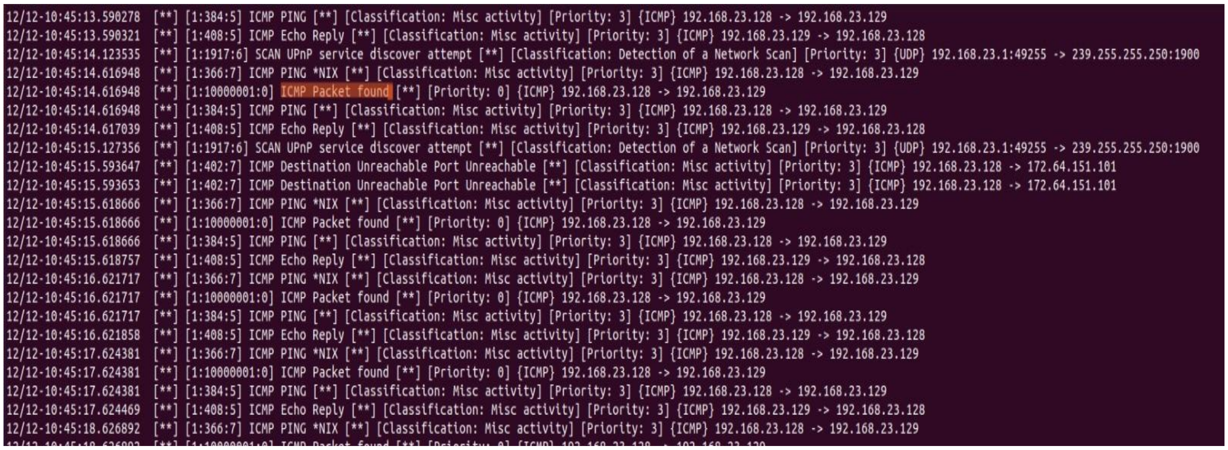


```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.23.129
PING 192.168.23.129 (192.168.23.129) 56(84) bytes of data.
64 bytes from 192.168.23.129: icmp_seq=1 ttl=64 time=1.11 ms
64 bytes from 192.168.23.129: icmp_seq=2 ttl=64 time=1.28 ms
64 bytes from 192.168.23.129: icmp_seq=3 ttl=64 time=1.33 ms
64 bytes from 192.168.23.129: icmp_seq=4 ttl=64 time=1.16 ms
64 bytes from 192.168.23.129: icmp_seq=5 ttl=64 time=0.458 ms
64 bytes from 192.168.23.129: icmp_seq=6 ttl=64 time=0.436 ms
64 bytes from 192.168.23.129: icmp_seq=7 ttl=64 time=0.767 ms
64 bytes from 192.168.23.129: icmp_seq=8 ttl=64 time=1.05 ms
64 bytes from 192.168.23.129: icmp_seq=9 ttl=64 time=1.13 ms
64 bytes from 192.168.23.129: icmp_seq=10 ttl=64 time=1.16 ms
64 bytes from 192.168.23.129: icmp_seq=11 ttl=64 time=0.558 ms
64 bytes from 192.168.23.129: icmp_seq=12 ttl=64 time=1.31 ms
64 bytes from 192.168.23.129: icmp_seq=13 ttl=64 time=1.11 ms
```

And in the target we execute the command :

Sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -I ens33

To see the reaction of snort.



```
12/12-10:45:13.590278  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:13.590321  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.129 -> 192.168.23.128
12/12-10:45:14.123535  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.23.1:49255 -> 239.255.255.250:1900
12/12-10:45:14.616948  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:14.616948  [**] [1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:14.616948  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:14.617039  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.129 -> 192.168.23.128
12/12-10:45:15.127356  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.23.1:49255 -> 239.255.255.250:1900
12/12-10:45:15.593647  [**] [1:402:7] ICMP Destination Unreachable Port Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 172.64.151.101
12/12-10:45:15.593653  [**] [1:402:7] ICMP Destination Unreachable Port Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 172.64.151.101
12/12-10:45:15.618666  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:15.618666  [**] [1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:15.618666  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:15.618757  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.129 -> 192.168.23.128
12/12-10:45:16.621717  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:16.621717  [**] [1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:16.621717  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:16.621858  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.129 -> 192.168.23.128
12/12-10:45:17.624381  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:17.624381  [**] [1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:17.624381  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:17.624469  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.129 -> 192.168.23.128
12/12-10:45:18.626892  [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.23.128 -> 192.168.23.129
12/12-10:45:18.626892  [**] [1:10000001:0] ICMP Packet found [**] [Priority: 0] {ICMP} 192.168.23.128 -> 192.168.23.129
```

Conclusion:

In conclusion, our cybersecurity project centered around the evaluation of Snort, an Intrusion Detection System (IDS), has provided valuable insights into its effectiveness in safeguarding network infrastructure. Through meticulous installation, configuration, and rule customization, we aimed to assess Snort's capabilities in detecting and responding to a variety of cyber threats. The project goals were met by simulating diverse attack scenarios and closely monitoring Snort's reaction.

Our findings highlight Snort's robust architecture, capable of efficiently processing network traffic and detecting known and unknown threats. The customization of rules, coupled with preprocessor configurations, demonstrated the flexibility of Snort in adapting to specific organizational needs. The analysis of response times, false positives, and adaptability to attack variations underscored Snort's position as a stalwart defender against an evolving threat landscape.

As we delved into rule configurations, especially focusing onTCP SYN and ICMP in this context, we witnessed how Snort can be fine-tuned to identify specific types of network activity. The ability to craft custom rules and implement thresholding provides organizations with a powerful tool to tailor their intrusion detection capabilities.

Our project emphasizes the importance of continuous maintenance, including rule updates and regular reviews, to ensure Snort remains resilient against emerging threats. Furthermore, the integration of Snort within a broader cybersecurity framework enhances its overall efficacy.