Configuring and Implementing Snort as an IDS/IPS System

(Ubuntu Server 22.04 LTS + Windows 11 Log Source)

Arjun U Menon

CICSA Batch : Sunday

TABLE OF CONTENTS

INTRODUCTION

In today's digital landscape, organizations face an ever-growing number of threats targeting their networks, ranging from malware and ransomware to unauthorized access and data exfiltration. To combat these threats, Intrusion Detection and Prevention Systems (IDS/IPS) play a pivotal role in ensuring network security by actively monitoring traffic, detecting malicious behavior, and taking appropriate actions.

An Intrusion Detection System (IDS) is primarily designed to detect and alert administrators about suspicious activity on a network. It passively observes network packets and logs threats for further analysis. On the other hand, an Intrusion Prevention System (IPS) not only detects threats but also actively takes steps to block or prevent them, such as dropping packets, resetting connections, or updating firewall rules.

Among the various IDS/IPS tools available, Snort stands out as one of the most powerful and widely adopted open-source network intrusion detection and prevention systems. Developed by Cisco, Snort is capable of real-time traffic analysis, protocol analysis, content searching/matching, and packet logging on IP networks. Its flexible rule-based language allows users to write custom rules for detecting a wide range of malicious activities, from simple port scans to complex exploits.

This report serves as a comprehensive guide to the configuration and deployment of Snort in a virtualized test environment, simulating real-world usage. It walks through the installation process, configuration steps, creation of custom detection rules, and validation techniques to ensure proper functioning. The report also highlights how Snort can be configured in both IDS (monitoring-only) and IPS (active prevention) modes, offering flexibility depending on organizational needs.

By the end of this document, readers will gain a solid understanding of how to deploy Snort for both detection and prevention purposes, how to customize it for specific security use cases, and how to integrate it with additional tools for effective alerting and log management.

OBJECTIVES

The primary goals of this project are:

- To introduce the core concepts of IDS and IPS systems.
- To provide practical experience in deploying Snort as a network security tool.
- To demonstrate how to install, configure, and fine-tune Snort for traffic analysis.
- To simulate real-world attacks and validate Snort's capability to detect and block threats.
- To create custom rules for monitoring specific network behaviours.
- To integrate Snort with other tools to enhance its detection and response capabilities.

VIRTUAL ENVIRONMENT SETUP

Host System Requirements: A 64-bit computer with at least 8 GB of RAM (16 GB recommended for smoother operation), a minimum of 100 GB of free disk space, and virtualization software such as VirtualBox (used in this project) or VMware Workstation/Player. Ensure that virtualization technology (VT- x for Intel or AMD-V for AMD) is enabled in the host system's BIOS/UEFI.

Virtual Machine Specifications:

· **Ubuntu Server 22.04 LTS: 2 virtual CPUs (vCPUs), 4 GB RAM, 40 GB virtual disk (dynamically allocated).**

· **Windows 11 Pro: 2 virtual CPUs (vCPUs), 4 GB RAM, 40 GB virtual disk (dynamically allocated, ensure TPM 2.0 and Secure Boot are enabled or bypassed for VM).**

Networking Configuration:

- Host-Only Adapter: Allows VMs to communicate with each other but isolates them from the internet.

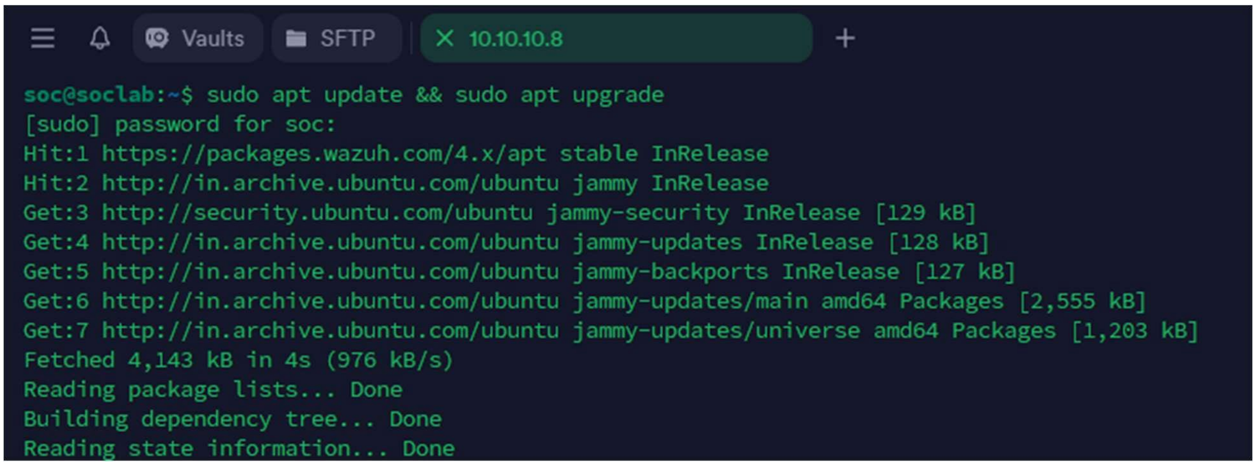
- Internal Network/Bridged Mode: Used for realistic testing scenarios where the Snort sensor can monitor all traffic between VMs.

SNORT INSTALLATION (UBUNTU 22.04 VIA QUICKSTART)

Step 1: Update the System

Ensure your Ubuntu server is up to date before starting the Wazuh installation:

```
sudo apt update && sudo apt upgrade -y
```



```
soc@soclab:~$ sudo apt update && sudo apt upgrade
[sudo] password for soc:
Hit:1 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,555 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,203 kB]
Fetched 4,143 kB in 4s (976 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

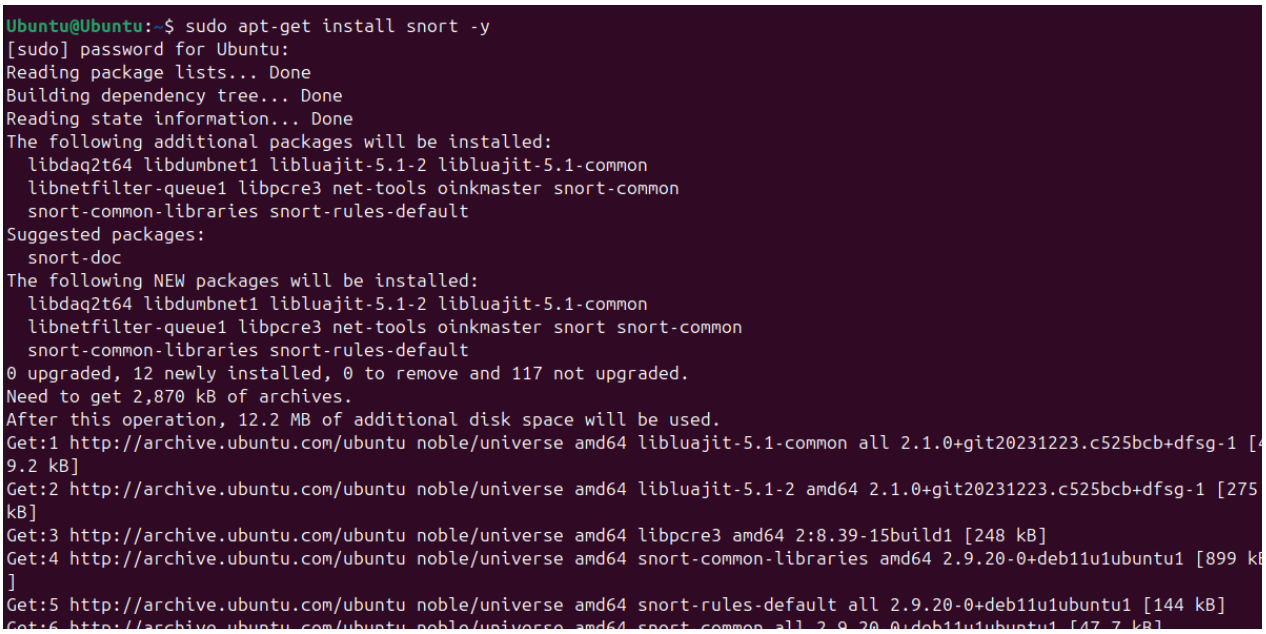
Step 2: Install Snort Data Acquisition (DAQ) Library

DAQ is a required component that provides an abstraction layer for packet I/O.

```
wget https://www.snort.org/downloads/snort/daq-2.X.tar.gz
tar -xvzf daq-2.X.tar.gz
cd daq-2.X
./configure && make && sudo make install
```

Step 3: Install the Snort

```
sudo apt-get install snort -y
```



```
Ubuntu@Ubuntu:~$ sudo apt-get install snort -y
[sudo] password for Ubuntu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libdaq2t64 libdumbnet1 libluajit-5.1-2 libluajit-5.1-common
  libnetfilter-queue1 libpcre3 net-tools oinkmaster snort-common
  snort-common-libraries snort-rules-default
Suggested packages:
  snort-doc
The following NEW packages will be installed:
  libdaq2t64 libdumbnet1 libluajit-5.1-2 libluajit-5.1-common
  libnetfilter-queue1 libpcre3 net-tools oinkmaster snort snort-common
  snort-common-libraries snort-rules-default
0 upgraded, 12 newly installed, 0 to remove and 117 not upgraded.
Need to get 2,870 kB of archives.
After this operation, 12.2 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 libluajit-5.1-common all 2.1.0+git20231223.c525bcb+dfsg-1 [4
9.2 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 libluajit-5.1-2 amd64 2.1.0+git20231223.c525bcb+dfsg-1 [275
kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/universe amd64 libpcre3 amd64 2:8.39-15build1 [248 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble/universe amd64 snort-common-libraries amd64 2.9.20-0+deb11u1ubuntu1 [899 kB
]
Get:5 http://archive.ubuntu.com/ubuntu noble/universe amd64 snort-rules-default all 2.9.20-0+deb11u1ubuntu1 [144 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/universe amd64 snort-common all 2.9.20-0+deb11u1ubuntu1 [47.7 kB]
```

INSTALLATION ON WINDOWS 11

Step 1: Download Snort for Windows:

Visit the official Snort website:
👉 https://www.snort.org/downloads

Create a Snort account and log in.

Download the latest Snort Windows binary (e.g., snort-2.9.x.x-win64.exe) or use the Snort tarball if binaries are not available.

Extract the file (if it's a .tar.gz) to a suitable directory like:

```
C:\Snort\
```

Step 2: Set Environment Variables:

Press Windows + S, type Environment Variables, and open Edit the system environment variables.
Under "System Properties" → Click Environment Variables.
In "System variables", find the Path variable → Click Edit → Click New.

Add:

```
C:\Snort\bin
```

Click OK on all windows to save and apply

SNORT CONFIGURATION

After installing Snort and its dependencies, the next step is to configure it for proper operation. This includes setting up directories, copying configuration files, and editing the main configuration file (snort.conf).

Step 1: Create Directory Structure:

```
sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /var/log/snort
sudo mkdir /usr/local/lib/snort_dynamicrules
```

- /etc/snort: Main configuration directory where Snort's core config files (e.g., snort.conf) are stored.
- /etc/snort/rules: Contains rule files, including local.rules, where custom rules are written.
- /var/log/snort: Default location for Snort's output logs and alerts.
- /usr/local/lib/snort_dynamicrules: Directory to hold Snort's dynamic rules (if any are used in future).

Step 2: Copy Default Configuration Files:

```
sudo cp etc/*.conf* /etc/snort
sudo cp etc/*.map /etc/snort
```

These commands copy default configuration and classification files from the Snort source directory (after compilation) to the proper configuration directory.

Files include:

- snort.conf: Main configuration file where Snort's behavior is defined.
- classification.config: Contains categories and priorities of alerts.
- reference.config: Maps rule references to external documentation.
- .map files: Help in decoding protocol messages and ports.

Without these files, Snort won't be able to process traffic or rules properly.

Step 3: Edit snort.conf:

```
sudo nano /etc/snort/snort.conf
```

The snort.conf file controls every aspect of how Snort behaves. This file must be edited carefully to:

a) **Set the Network Variables**
   var HOME_NET 192.168.56.0/24
b) Define Rule Paths

```
var RULE_PATH /etc/snort/rules
include $RULE_PATH/local.rules
```

c) Ensure output plugin:

```
output alert_fast: stdout
```

CREATING AND MANAGING SNORT RULES

One of Snort's core strengths is its rule-based detection system. These rules define what network behavior should be observed and how Snort should respond. Rules can detect anything from ICMP pings to suspicious payloads and port scans.

Basic Rule Structure

```
action protocol src_ip src_port -> dest_ip dest_port (options)
```

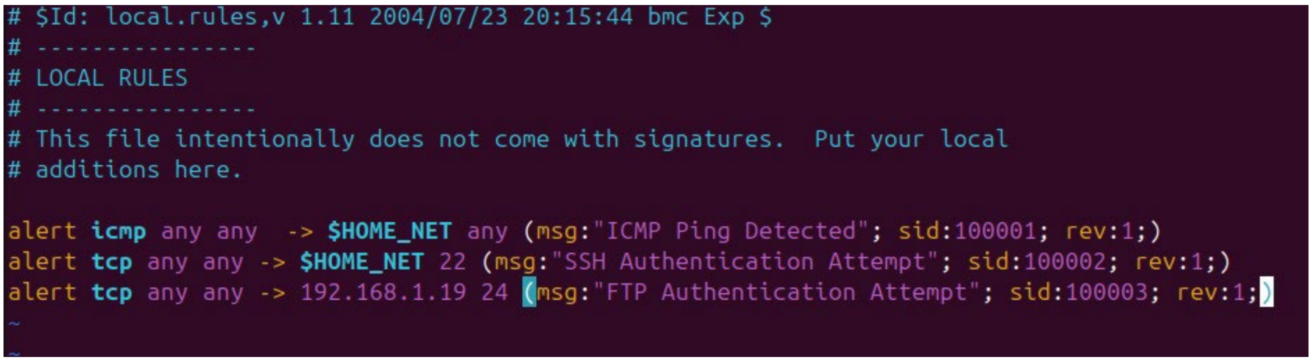
Component	Description
action	What Snort should do: alert, log, pass, drop, etc.
protocol	Protocol to watch: tcp, udp, icmp, or ip
src_ip	Source IP address (any, specific IP, or variable like $EXTERNAL_NET)
src_port	Source port (can be any, single port, or range)
-> / <>	Direction of traffic (-> for unidirectional, <> for bidirectional)
dest_ip	Destination IP address (can be a variable like $HOME_NET)
dest_port	Destination port (can be any or a specific port)
(options)	Additional details in parentheses: message, rule ID, payload patterns, etc.

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

Breakdown:

- alert → Action: Trigger an alert.
- icmp → Protocol being monitored.
- any any → Source IP and port (match any).
- $HOME_NET any → Destination is the internal network (Snort variable).
- msg → Message to show when rule is triggered.

- sid → Snort Rule ID (must be unique).
- rev → Revision number of the rule.



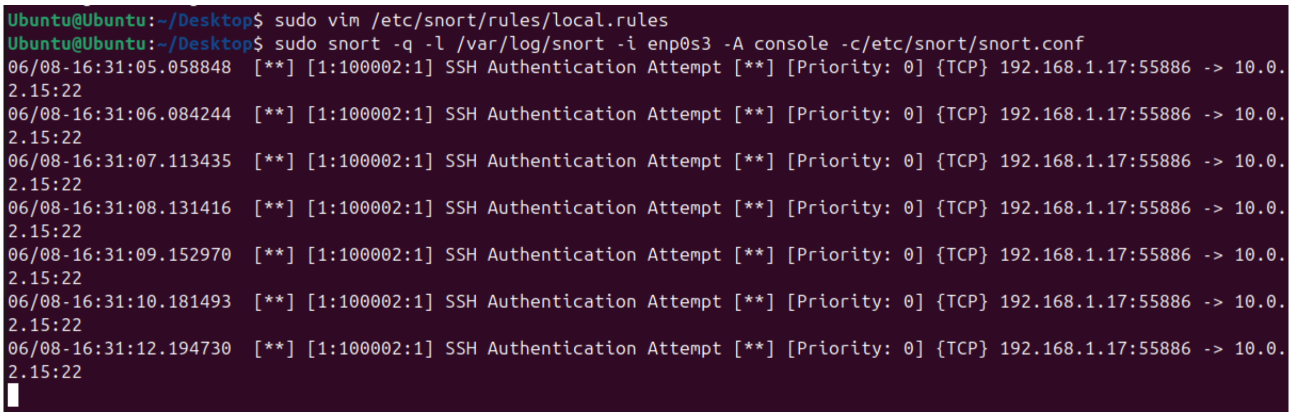
```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

alert icmp any any  -> $HOME_NET any (msg:"ICMP Ping Detected"; sid:100001; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Attempt"; sid:100002; rev:1;)
alert tcp any any -> 192.168.1.19 24 (msg:"FTP Authentication Attempt"; sid:100003; rev:1;)
~
~
```

Steps to Add Custom Rules

1. Open the Custom Rules File

Snort loads rules from specific files defined in snort.conf. The local.rules file is typically used for user-defined rules.

```
sudo nano /etc/snort/rules/local.rules
```



```
Ubuntu@Ubuntu:~/Desktop$ sudo vim /etc/snort/rules/local.rules
Ubuntu@Ubuntu:~/Desktop$ sudo snort -q -l /var/log/snort -i enp0s3 -A console -c/etc/snort/snort.conf
06/08-16:31:05.058848  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55886 -> 10.0.
2.15:22
06/08-16:31:06.084244  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55886 -> 10.0.
2.15:22
06/08-16:31:07.113435  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55886 -> 10.0.
2.15:22
06/08-16:31:08.131416  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55886 -> 10.0.
2.15:22
06/08-16:31:09.152970  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55886 -> 10.0.
2.15:22
06/08-16:31:10.181493  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55886 -> 10.0.
2.15:22
06/08-16:31:12.194730  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55886 -> 10.0.
2.15:22
```

2. Add a Custom Rule

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

Ensure sid numbers for custom rules are above 1000000 to avoid conflicts with official rules.

3. Save and Exit
   Press:

- CTRL + O to write (save)
- ENTER to confirm
- CTRL + X to exit

Testing Rule Syntax

```
sudo snort -T -c /etc/snort/snort.conf
```

RUNNING SNORT AS AN INTRUSION DETECTION SYSTEM (IDS)

After configuring Snort and adding rules, the next step is to deploy Snort in different modes to observe, log, and analyze network traffic. Snort can be run in three primary modes:

1. Sniffer Mode – simple packet display
2. Packet Logger Mode – logs packets to files
3. IDS Mode with Rules – detects and alerts based on rule violations

Each mode is essential during different phases of deployment, from testing to real-time threat monitoring.

1. Sniffer Mode

In Sniffer Mode, Snort acts like a traditional packet sniffer (e.g., tcpdump or Wireshark). It simply reads packets from the specified interface and prints basic information about them to the console.

```
sudo snort -v -i eth0
```

Explanation:

- -v enables verbose output (prints packet headers).
- -i eth0 specifies the network interface (replace eth0 with your interface, like ens33 or wlan0).

Output:
This mode displays real-time packet headers on the terminal:

- Source and destination IP addresses
- Protocols used (TCP, UDP, ICMP)
- Packet size and flags

2. Packet Logger Mode

This mode logs raw packets into files for further analysis. It is useful for storing traffic data to investigate anomalies, conduct forensics, or analyze unknown payloads later.

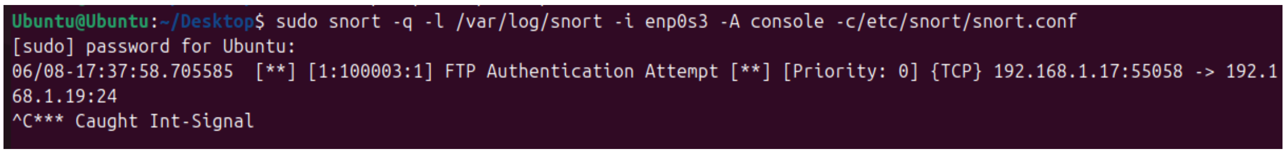
```
sudo snort -dev -i eth0 -l /var/log/snort
```

Explanation:

- -d: Dump application layer data (payloads).
- -e: Show Ethernet headers.

- -v: Verbose mode.
- -i eth0: Interface to monitor.
- -l /var/log/snort: Directory where logs will be saved.

Snort will create subdirectories and save captured packet data in formats like snort.log.*.



```
Ubuntu@Ubuntu:~/Desktop$ sudo snort -q -l /var/log/snort -i enp0s3 -A console -c/etc/snort/snort.conf
[sudo] password for Ubuntu:
06/08-17:37:58.705585  [**] [1:100003:1] FTP Authentication Attempt [**] [Priority: 0] {TCP} 192.168.1.17:55058 -> 192.1
68.1.19:24
^C*** Caught Int-Signal
```

3. IDS Mode with Rules (Detection Mode)

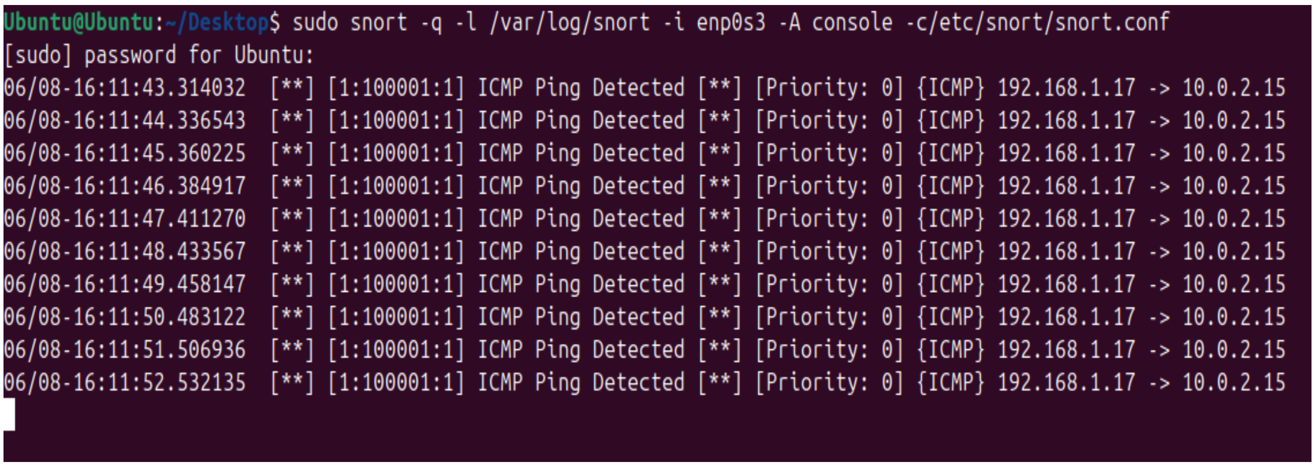
This is Snort's full-fledged intrusion detection mode. Snort monitors traffic and triggers alerts based on defined rules.

```
sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
```

Explanation:

- -A console: Print alerts to the terminal.
- -q: Quiet mode (suppresses banner and summary).
- -c /etc/snort/snort.conf: Use the main configuration file.
- -i eth0: Network interface.

If an event matches any rule (from local.rules or default rules), Snort will print alert messages directly to the terminal.



```
Ubuntu@Ubuntu:~/Desktop$ sudo snort -q -l /var/log/snort -i enp0s3 -A console -c/etc/snort/snort.conf
[sudo] password for Ubuntu:
06/08-16:11:43.314032  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:44.336543  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:45.360225  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:46.384917  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:47.411270  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:48.433567  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:49.458147  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:50.483122  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:51.506936  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
06/08-16:11:52.532135  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.1.17 -> 10.0.2.15
```

RUNNING SNORT AS AN INTRUSION PREVENTION SYSTEM (IPS)

While Snort is traditionally known for its IDS (Intrusion Detection System) capabilities, it can also be configured as an Intrusion Prevention System (IPS). In IPS mode, Snort not only detects malicious traffic but also takes action to block or drop the packets in real time, thereby actively protecting the network.

To run Snort in IPS mode on a Linux environment, we need to integrate it with iptables using the NFQUEUE feature, which allows Snort to inspect and control packets before they reach their destination.

In IPS mode, packets are queued by the Linux Netfilter firewall (iptables) and handed over to Snort via a queue mechanism. Snort inspects these packets and, based on rule actions like drop, decides whether to allow or block them.

Step-by-Step Configuration of Snort as an IPS

Step 1: Enable IP Forwarding

IP forwarding allows your system to route packets between networks — an essential feature when using Snort as a bridge or inline device.

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

What This Does:

- This command enables packet forwarding in the Linux kernel.
- Without this, packets will not traverse through the system.

Note: To make this change permanent, edit /etc/sysctl.conf and uncomment/add:

```
net.ipv4.ip_forward=1
sudo sysctl -p
```

Step 2: Configure iptables to Use NFQUEUE

This step sets up the **firewall rules** to redirect incoming packets into a queue (queue number 0) that Snort can monitor and process.

```
sudo iptables -I INPUT -j NFQUEUE --queue-num 0
```

- -I INPUT: Inserts the rule at the top of the INPUT chain.
- -j NFQUEUE: Directs matching packets into the NFQUEUE.
- --queue-num 0: Assigns the queue number for Snort to use.

Step 3: Run Snort in Inline Mode Using NFQUEUE

Now run Snort in inline (IPS) mode, specifying the --daq nfq interface and the correct queue number.

```
sudo snort -Q --daq nfq --daq-var queue=0 -c
/etc/snort/snort.conf
```

- -Q: Enables inline (IPS) mode.
- --daq nfq: Uses the NetFilter Queue DAQ module.
- --daq-var queue=0: Specifies the queue number to match with iptables.
- -c: Path to the Snort configuration file.

Step 4: Modify Rules to Use drop Instead of alert

To actively block traffic, you must use the **drop** keyword in your rules. While alert only logs the incident, drop **blocks** the packet and logs the event.

```
drop tcp any any -> $HOME_NET 22 (msg:"SSH Blocked"; sid:1000002;
rev:1;)
```

- drop: Blocks the matching packet and logs an alert.
- tcp any any -> $HOME_NET 22: Match TCP traffic trying to access port 22 (SSH) on the internal network.
- msg: Log message for identification.
- sid: Unique Snort rule identifier.
- rev: Rule revision number.

TESTING AND VALIDATION

After configuring Snort as an IDS or IPS, it's crucial to validate whether it functions as expected. This involves simulating real-world attacks using recognized penetration testing tools and verifying whether Snort detects and/or blocks them.

Testing helps ensure:

- Snort is actively monitoring the correct interfaces.
- Detection and drop rules are functioning properly.
- Logs are generated and stored in the correct location.
- Alerts are accurate, with minimal false positives or negatives.

Tools Used for Testing:

To simulate different types of attacks and observe Snort's behavior, we use the following industry-standard tools:

1. Nmap – Port Scanning

Nmap (Network Mapper) is a widely used tool for **network discovery and security auditing**. It can scan a host for open ports and services.

```
nmap -sS 192.168.56.101
```

- -sS: Performs a TCP SYN (stealth) scan.
- 192.168.56.101: Target IP address (change this to your monitored host IP).

Snort should generate an alert indicating a possible port scan. You can define or use existing rules that detect TCP SYN floods or scans.

```
sudo tail -f /var/log/snort/alert
```

2. Nikto – Web Server Vulnerability Scanning

Nikto is an open-source tool for scanning web servers for known vulnerabilities, outdated software, and misconfigurations.

```
nikto -h http://192.168.56.101
```

- -h: Specifies the target host.
- Nikto sends crafted HTTP requests to test for weak headers, XSS vulnerabilities, insecure cookies, etc.

If Snort is monitoring the HTTP port (port 80 or 443), it should detect suspicious or malformed requests and generate alerts.

Test	Tool	Expected Snort Behavior	Verified
Port Scan	Nmap	Detects and logs SYN scans	✅ / ❌
Web Scan	Nikto	Detects HTTP attacks	✅ / ❌
Exploit	Metasploit	Detects reverse shell or exploit	✅ / ❌

PERFORMANCE TUNING AND OPTIMIZATION

Deploying Snort in a production or research environment requires not only accurate detection but also efficient performance. As network traffic and rule complexity increase, Snort may experience latency, packet loss, or false positives if not properly optimized.

This section outlines various strategies for tuning Snort's performance, ensuring real-time monitoring while maintaining system stability.

1. Disable Unused or Irrelevant Rules

Snort's default rule sets (such as community rules) include thousands of signatures for various protocols, exploits, and attack vectors. Not all of these apply to every environment.

Step 1: Open the Snort configuration file:

```
sudo nano /etc/snort/snort.conf
```

Step 2: Locate the section where rules are included. It typically looks like:

```
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/malware.rules
```

Step 3: Comment out unnecessary rule sets by adding a #:

```
#include $RULE_PATH/attack-responses.rules
#include $RULE_PATH/malware.rules
```

Benefits:

- Reduces memory usage and CPU overhead.
- Speeds up rule evaluation.
- Prevents unnecessary alerts.

2. Create and Use Custom Rules

Instead of loading entire community or enterprise rulesets, you can write **targeted rules** specific to your environment. This ensures relevance and better performance.

Step 1: Open local rule file:

```
sudo nano /etc/snort/rules/local.rules
```

Step 2: Write tailored rules for services and assets that matter in your network. For example:

```
alert tcp any any -> $HOME_NET 3306 (msg:"MySQL Access Detected";
sid:1000010;)
alert tcp any any -> $HOME_NET 22 (msg:"SSH Access Detected";
sid:1000011;)
```

Step 3: Test rule configuration:

```
sudo snort -T -c /etc/snort/snort.conf
```

Step 4: Monitor Snort performance and log size after implementation.

- Avoids irrelevant detections.
- Reduces noise in alert logs.
- Improves response time and threat focus.

CONCLUSION

In the evolving landscape of cybersecurity, proactive monitoring and threat detection have become indispensable for safeguarding digital infrastructure. This project has demonstrated the complete lifecycle of configuring and deploying **Snort**, a powerful open-source Intrusion Detection and Prevention System (IDS/IPS), within a controlled virtual lab environment.

Throughout this report, we systematically explored the installation, configuration, rule creation, and real-time testing of Snort to understand how it operates as both an IDS and an IPS. Beginning with a well-planned virtual environment setup, we ensured isolation and safety while replicating real-world scenarios. By installing Snort from source and carefully managing its dependencies, we gained insights into its modular architecture and the flexibility it offers for customization.

The configuration phase emphasized the importance of defining accurate network variables, setting up rule directories, and understanding how Snort preprocessors and output plugins function. Writing and implementing custom rules was a pivotal component of this project, as it allowed us to define specific conditions for traffic inspection, alert generation, and packet blocking. We also explored the nuances of running Snort in various modes — sniffer, logger, and inline — showcasing its capability to both detect and prevent malicious activity.

Real-time attack simulations using tools like **Nmap**, **Nikto**, and **Metasploit** helped validate Snort's effectiveness and responsiveness in detecting common threats, such as port scanning, brute-force attempts, and ICMP-based reconnaissance. These tests reinforced the practical applicability of Snort in defending networks against unauthorized access and exploitation attempts.

Furthermore, we addressed performance tuning techniques and integration options with visualization tools like **BASE** or security-focused distributions like **Security Onion**, which can enhance Snort's usability in enterprise environments.

REFERENCES

The Snort Project – Official Documentation
Cisco Systems. (n.d.). *Snort Documentation*. Retrieved from:
https://docs.snort.org

Snort Official Website
Cisco Systems. (n.d.). *Snort Intrusion Detection and Prevention System*. Retrieved from:
https://www.snort.org

Snort Rules Documentation
Cisco Systems. (n.d.). *Writing Snort Rules*. Retrieved from:
https://snort.org/documents

Snort 2.9.20 Installation Guide on Ubuntu
Linux Hint. (n.d.). *How to Install Snort on Ubuntu*. Retrieved from:
https://linuxhint.com/install_snort_ubuntu/

Barnyard2 Guide for Snort Logging
Offensive Security. (n.d.). *Using Barnyard2 with Snort*. Retrieved from:
https://tools.kali.org/information-gathering/snort

The ELK Stack for Network Security Monitoring
Elastic.co. (n.d.). *Getting Started with ELK Stack*. Retrieved from:
https://www.elastic.co/what-is/elk-stack

Nmap Official Documentation
Insecure.org. (n.d.). *Nmap: Network Scanning*. Retrieved from:
https://nmap.org/book/man.html

Nikto Web Scanner
CIRT.net. (n.d.). *Nikto2 Documentation*. Retrieved from:
https://cirt.net/Nikto2

Metasploit Framework Documentation
Rapid7. (n.d.). *Metasploit Unleashed*. Retrieved from:
https://docs.rapid7.com/metasploit/

Security Onion Project
Security Onion Solutions. (n.d.). *Security Onion Documentation*. Retrieved from:
https://docs.securityonion.net