

**Individual Assignment**

|  |  |  |
| --- | --- | --- |
| **Module Code** | **:** | **IMPLEMENTATION OF SECURE SYSTEM CT128-3-2-ISC** |
| **Intake Code** | **:** | **APD2F2409CS(CYB)** |
| **Lecturer Name** | **:** | **Ms. Noris Ismail** |
| **Hand In Date** | **:** | **3rd March 2025** |
| **Hand Out Date** | **:** | **31st May 2025** |
| **Lab No** | **:** | **Lab-2** |

|  |  |
| --- | --- |
| **Student Name** | **Student ID** |
| Khegai Olga | TP069365 |

**Table of Contents**

[Abstract 5](#_Toc199619978)

[1.0 Introduction 5](#_Toc199619979)

[1.1 Overview of Network Intrusion Detection 5](#_Toc199619980)

[1.2 Importance of Detecting Port Scanning and Brute Force Attacks 6](#_Toc199619981)

[2.0 Selected Focus Area: Snort as IDS 6](#_Toc199619982)

[2.1 What is Snort? 6](#_Toc199619983)

[2.2 How Snort Works 7](#_Toc199619984)

[2.3 Relevance to Modern Threats 7](#_Toc199619985)

[3.0 Threats Overview 8](#_Toc199619986)

[3.1 Port Scanning (Nmap, Types of Scans) 8](#_Toc199619987)

[3.2 Brute Force Attacks (Hydra, Password Guessing) 9](#_Toc199619988)

[4.0 Tools and Environment 9](#_Toc199619989)

[4.1 Kali Linux 9](#_Toc199619990)

[4.2 Snort 10](#_Toc199619991)

[4.3 VMware Network Topology 10](#_Toc199619992)

[4.4 Nmap, Hydra, Metasploit 10](#_Toc199619993)

[5.0 Implementation 11](#_Toc199619994)

[5.1 Setting Up the Network in VMware 11](#_Toc199619995)

[5.2 Creating the Second Kali VM via Cloning 14](#_Toc199619996)

[5.3 Installing Snort 17](#_Toc199619997)

[5.4 Creating and Testing Snort Rules 18](#_Toc199619998)

[5.5 Simulating Port Scan with Nmap 21](#_Toc199619999)

[5.6 Simulating Brute Force with Hydra or Metasploit 22](#_Toc199620000)

[5.7 Capturing and Analyzing Logs 25](#_Toc199620001)

[5.7.1 ICMP Ping Simulation – Log Analysis 25](#_Toc199620002)

[5.7.2 Nmap Port Scan Simulation – Log Analysis 26](#_Toc199620003)

[5.7.3 Hydra Brute Force Simulation – Log Analysis 27](#_Toc199620004)

[6.0 Testing and Evaluation 28](#_Toc199620005)

[6.1 Detection Accuracy 28](#_Toc199620006)

[6.2 False Positives / Limitations 28](#_Toc199620007)

[6.3 Overall Effectiveness 29](#_Toc199620008)

[7.0 Conclusion 29](#_Toc199620009)

[8.0 Reflection 29](#_Toc199620010)

[9.0 References 30](#_Toc199620011)

**Table of Figures**

[Figure 1: Intrusion Detection System (IDS) 5](#_Toc199615863)

[Figure 2: Snort Tool 6](#_Toc199615864)

[Figure 3: Port Scanning 8](#_Toc199615865)

[Figure 4: Brute Force Attack 9](#_Toc199615866)

[Figure 5: VMnet8 NAT Network Configuration 11](#_Toc199615867)

[Figure 6: NAT Settings for VMnet8 12](#_Toc199615868)

[Figure 7: DHCP Settings for VMnet8 12](#_Toc199615869)

[Figure 8: Configuring Network on Kali Attacker 13](#_Toc199615870)

[Figure 9: Static IP for Kali Attacker 13](#_Toc199615871)

[Figure 10: Restart the Network and Check IP 13](#_Toc199615872)

[Figure 11: Cloning Kali Linux 14](#_Toc199615873)

[Figure 12: Setting the Hostname on Kali Clone VM 14](#_Toc199615874)

[Figure 13: /etc/hosts command 14](#_Toc199615875)

[Figure 14: /etc/hosts update 15](#_Toc199615876)

[Figure 15: Reboot 15](#_Toc199615877)

[Figure 16: Kali Snort VM 15](#_Toc199615878)

[Figure 17: System ID reset 15](#_Toc199615879)

[Figure 18: Configuring Network on Kali Snort Command 15](#_Toc199615880)

[Figure 19: Static IP for Kali Snort 16](#_Toc199615881)

[Figure 20: IP Check on Kali Snort 16](#_Toc199615882)

[Figure 21: Ping Snort on Attacker 16](#_Toc199615883)

[Figure 22: Ping Attacker on Snort 17](#_Toc199615884)

[Figure 23: Installation of Snort 17](#_Toc199615885)

[Figure 24: Verifying version 18](#_Toc199615886)

[Figure 25: Creating Snort Directories 18](#_Toc199615887)

[Figure 26: Creating Rules File 18](#_Toc199615888)

[Figure 27: Rule scenario 18](#_Toc199615889)

[Figure 28: Running Snort with the Rule 19](#_Toc199615890)

[Figure 29: Ping from Attacker 20](#_Toc199615891)

[Figure 30: Result of Snort Alert 20](#_Toc199615892)

[Figure 31: Configuring Rules for Snort 21](#_Toc199615893)

[Figure 32: Second Rule Insertion 21](#_Toc199615894)

[Figure 33: Launching Snort 21](#_Toc199615895)

[Figure 34: Nmap Attack 21](#_Toc199615896)

[Figure 35: Snort Detection of Nmap Attack 22](#_Toc199615897)

[Figure 36: local.rules 22](#_Toc199615898)

[Figure 37: Adding the Rule to trigger brute force attacks 23](#_Toc199615899)

[Figure 38: Snort 23](#_Toc199615900)

[Figure 39: Hydra installation 23](#_Toc199615901)

[Figure 40: Attack with Hydra 24](#_Toc199615902)

[Figure 41: Snort Detection 24](#_Toc199615903)

[Figure 42: ISMP Analysis 25](#_Toc199615904)

[Figure 43: TCP Analysis 26](#_Toc199615905)

[Figure 44: SSH Analysis 27](#_Toc199615906)

# Abstract

**Using IDS (Snort) to Detect Port Scanning and Brute Force Attacks.**

Cyberattacks of various types - from port scanning to brute force attempts - are targeted at network infrastructures where they are interconnected and online. This is normally the first stage of a larger intrusion. (Chandran, 2023) The goal of most attackers is to compile information in order to exploit it - vulnerabilities and/or credentials - to gain access to systems and information. As the purpose of this project, Snort, an open-source Intrusion Detection System (IDS), will be installed to detect the activities against simulated threat against a virtualized network. The simulated network included Kali Linux as the attacker and a simulated target machine protected by Snort. Using Nmap for port scanning and Hydra for brute force attack, I will attempt to scan and brute force attack into the target machine. (Tokyoneon, 2018) I learned to collect and log the malicious activities and review Snort alerts, reviewing Snort logs and analyzing detection capabilities. The demonstration illustrates how IDS systems such as Snort can provide visibility into networks and a potential defense layer against common cyber-attack vectors. (Borikar, 2024)

# 1.0 Introduction

## 1.1 Overview of Network Intrusion Detection

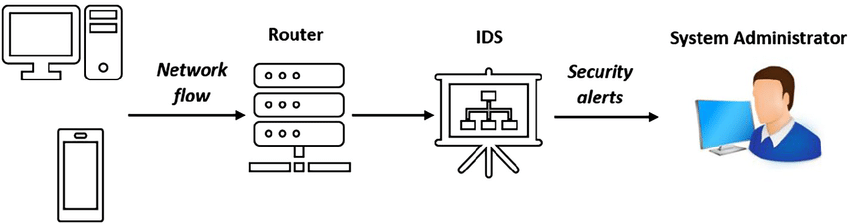


Figure 1: Intrusion Detection System (IDS)

The rapid increase in digital infrastructure means that organizations are exposed to more cyber threats targeting their network communications and resources. One of the most common attack vectors are port scanning and brute force attacks, which are typical methods during the reconnaissance and exploitation stages of a cyberattack. (Chandran, 2023) These attack vectors allow the bad actor to find open ports, services, and weak authentication in order to gain unwanted access.

Intrusion Detection Systems (IDS) have become a foundational piece in modern cybersecurity method. An IDS is a general-purpose cybersecurity solution which records and monitors network traffic in real-time for suspicious activities and generates alerts through a set of conditions defined by either rules or behaviors. Snort, an open-source signature-based IDS is one of the most used IDS solutions in enterprise and education environments, able to do packet logging and real-time analysis of traffic. (Borikar, 2024)

## 1.2 Importance of Detecting Port Scanning and Brute Force Attacks

Port scanning and brute force attacks are so easily overlooked because they serve as the foundation for more advanced intrusions. Time is critical in detecting port scans and brute force attacks. Once detected, administrators have ample opportunity to prevent lateral movement, privilege escalations and/or data breaches. Port scans will disclose valuable information about a network topology and its exposed services, as a proper brute force attack can provide a tricky user credentials leading to unauthorized access.

If an attack is detected in real-time, network administrators can react to terminate that malicious actor and help structure their vulnerable systems. Snort is a lightweight yet powerful tool that can be useful as a monitoring system for critical alerts, if configured correctly. (Tokyoneon, 2018) This project proposes that Snort can exists in a virtual lab environment used to detect these types of specific intrusions, strengthen security posture, and provide valuable insight into intrusion prevention.

# 2.0 Selected Focus Area: Snort as IDS

## 2.1 What is Snort?



Figure 2: Snort Tool

Snort is an open-source Intrusion Detection System (IDS) created by Martin Roesch in 1998 and maintained by Cisco. (Borikar, 2024) Snort is designed to perform real-time traffic analysis and packet logging against Internet Protocol (IP) networks. Snort allows administrators to define the traffic patterns they would like to review while detecting malicious or suspicious traffic. The program has three modes of operation; sniffer mode, packet logger mode, network intrusion detection mode — the last mode will hold the most relevance for this project.

Snort is particularly useful because of its versatility. For example, Snort has the capability to detect several kinds of attacks such as buffer overflows, port scans, stealth scans, CGI attacks, SMB probes, and much more. Its rule-based design allows for updates and alterations as attacks change over time.

## 2.2 How Snort Works

Snort was developed by coding software that would analyze network traffic by examining every packet against a set of rules that define suspicious behavior. When a match is found, Snort would generate an alert and log the event. The detection engine uses a rule set that incorporates both signature-based and protocol-based analysis which allows Snort to look for known attacks and other suspicious behaviors.

Snort is a modular-based package made up of five main components: packet decoder, preprocessor, detection engine, logging and alerting, and output modules. It's modular design allows Snort to integrate easily with external programs and operate well in both small scale environments and enterprise environments. (Chandran, 2023)

## 2.3 Relevance to Modern Threats

The increasing number of reconnaissance style attacks, such as port scanning and brute force logins, emphasizes the need to implement monitoring software, such as Snort. These reconnaissance practices are usually the precursor to a more complex attack. Without monitoring and alerting, reconnaissance practices such as these can exist without notice until significant damage occurs.

Snort can allow cybersecurity students and practitioners to simulate real-world attack patterns and still detect them in a controlled environment. In this project, Snort will be implemented in a controlled VMware network to detect two threat events, Nmap port scans and Hydra brute force logins. This will allow Snort to demonstrate it can recognize unauthorized access attempts and remind users of the need to monitor a network, as the scans would otherwise go unnoticed. (Tokyoneon, 2018)

# 3.0 Threats Overview

## 3.1 Port Scanning (Nmap, Types of Scans)

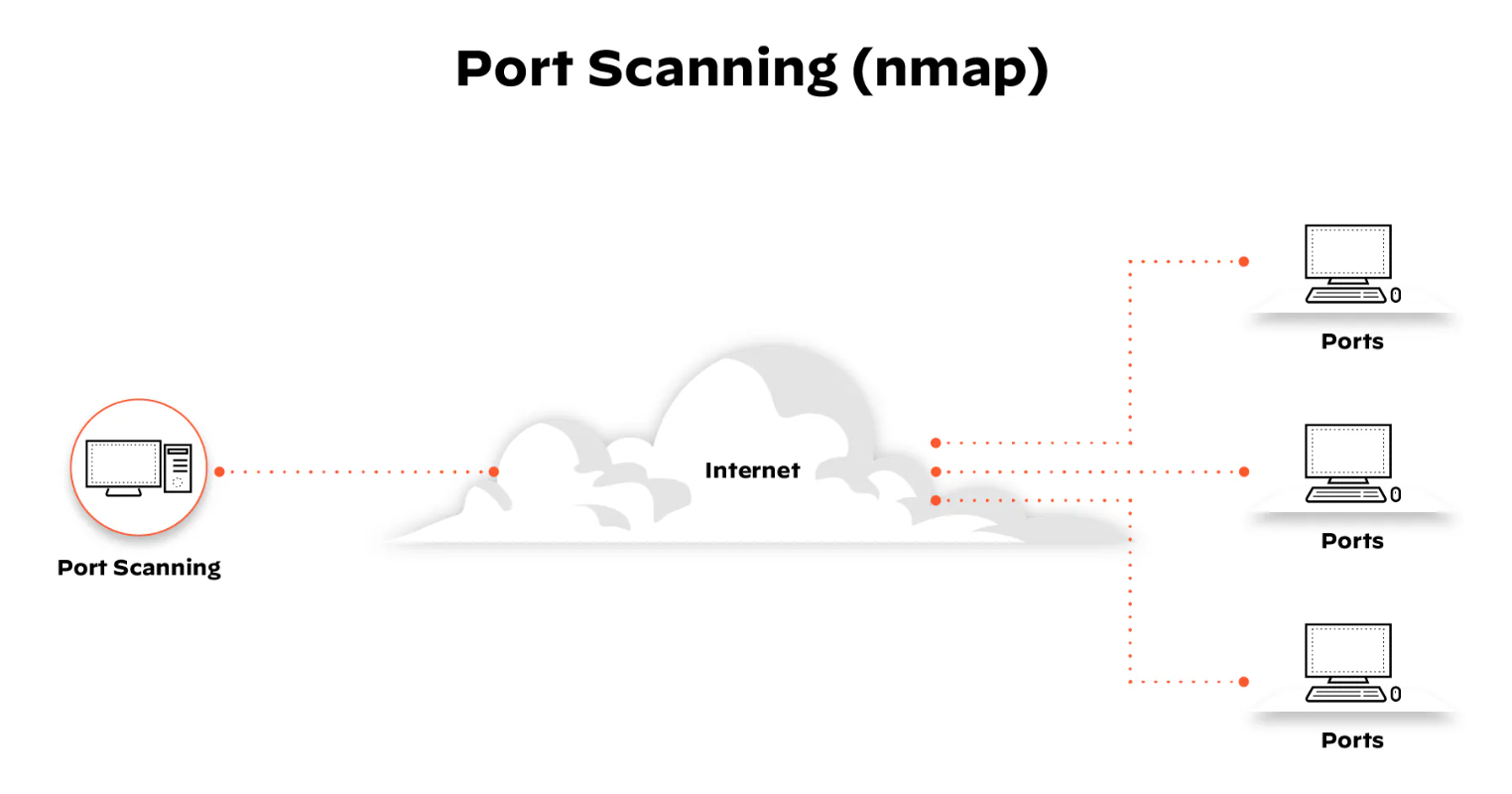


Figure 3: Port Scanning

Port scanning is a typical reconnaissance method used by attackers to find open ports and services on a targeted machine. Each open port usually corresponds to a specific service or application that may be exposed. Attackers can use programs like Nmap to perform a variety of scans, including SYN scan, TCP connect scan, and stealth scan, to gain as much information as possible about the target system without raising alarms. (Layne, 2025)

Port scanning is not malicious in and of itself, but it could lead to exploitation and is a sign of possible intrusion. A clean scan shows open, filtered, and closed ports, and which services are active and available. For administrators, detecting these scans in real-time is important because they can immediately respond to this behavior and block attacks.

This assignment has the potential for Nmap to be used within a Kali Linux virtual machine to perform SYN scans against a target VM. The purpose of this exercise is to determine if Snort can detect the scan behavior and manipulate the detection to log the attempt via configured rules.

## 3.2 Brute Force Attacks (Hydra, Password Guessing)

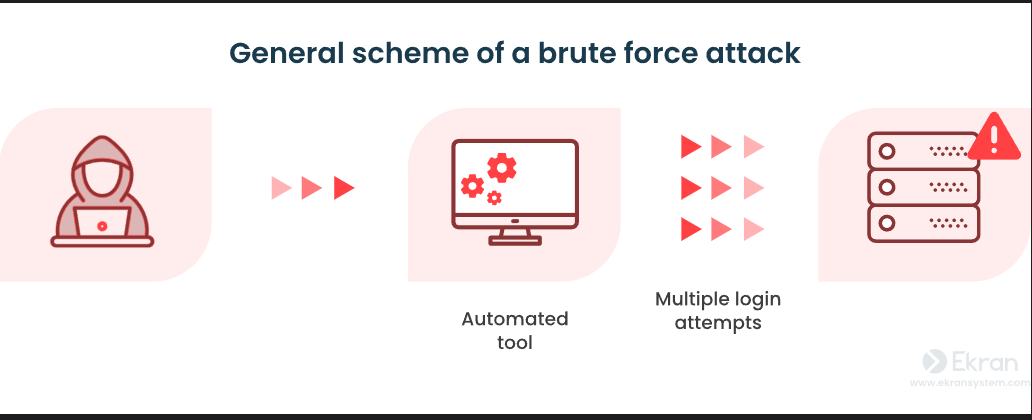


Figure 4: Brute Force Attack

Brute force attacks are a method of trying a lot of passwords, automatically, to break into a system. This type of attack is usually on services like SSH, FTP, or other login service. For example, tools such as Hydra or Metasploit can force credentials in a very short time. (Mitchell, 2024)

While brute force attacks are simple attacks, they can be effective for systems that have weak or default passwords. An attacker just needs to guess the password correctly, and then can simply escalate privileges, install malware, or exfiltrate sensitive data. That's why brute force attempts are risk classified as high, especially programs executed against administrative accounts.

In this project, the brute force simulation will occur with Hydra against a login service (e.g., SSH or FTP) on the victim machine. Then Snort will have some rules added to it and run to see if the rapid login attempts against a single source IP are detected and alerts can be created for analysis.

# 4.0 Tools and Environment

This project employed a simulated network lab environment with VMware Workstation Pro, to perform offensive security and defensive security techniques in a controlled environment. The tools were deployed within this environment to simulate real world attacks and be able to see the detection capabilities.

## 4.1 Kali Linux

Kali Linux is a Debian-based Linux distribution focused on penetration testing and ethical hacking. Kali has numerous pre-installed tools such as Nmap, Hydra, and Metasploit, to start recon and brute force attacks and was utilized here as the attacker VM, from which the port scanning and brute force attack were launched.

## 4.2 Snort

Snort is a root open-source Intrusion Detection System (IDS) that has the capacity to perform real-time packet analysis and act as a network-based threat detection. For this project, Snort was installed onto the victim VM and customized with specific rules to detect specific attacks, such as, port scanning attempts and brute force login attempts. Snort’s alerting and logging capabilities were utilized in the evaluation of the detection capability of the system

## 4.3 VMware Network Topology

VMware Workstation Pro was used to create and run a number of virtual machines in a single host. It provides control over the network configuration (such as NAT and host-only networks), allowing an attacker and victim machine to communicate securely in an isolated virtual environment. This configuration isolates any actions in the virtual network environment from the external network, and poses no risk to real systems

## 4.4 Nmap, Hydra, Metasploit

Nmap (Network Mapper) is powerful, open-source tool for network discovery and security auditing. Nmap was used in this project for simulating port scans attacks originating from the Kali Linux VM, and identifying which services on the target machine were open or vulnerable. Different scan types (e.g., SYN scan) were tested to consider Snort’s ability to detect different scanning methods.

Hydra is a fast and flexible online login cracker that is utilized for brute force attacks against network services such as SSH, FTP and HTTP. It was used in this project for simulating unauthorized login attempts against the victim machine. These unauthorized login attempts were monitored by Snort for evaluating if such brute force behavior could be detected and alerted.

While Metasploit can certainly execute brute force attacks with some of its auxiliary modules, Hydra was used for this project because it is straightforward to use, efficient, and provides clear logs. Hydra was specifically designed to crack logins and provides very concise logs that match with Snort's capabilities for detection. Metasploit, on the other hand, is a comprehensive exploitation framework which adds extra layers of noise and complexity that will reduce the precision of the IDS alerts. For the demonstration of brute force activity in a controlled lab environment, Hydra was identified as a better tool to provide analysis. (Network Security: Analyzing Metasploit and Hydra Tools, n.d.)

# 5.0 Implementation

## 5.1 Setting Up the Network in VMware

To establish a secure and controlled testing environment for network intrusions, two Kali Linux virtual machines were created in VMware Workstation Pro. One virtual machine acted as the attacker and the other served as the Snort IDS-enabled victim.

The two virtual machines were connected to VMware's default NAT network (vmnet8) which made traffic monitoring and manual configuration of IP addresses more straightforward. The NAT network was reconfigured to use a custom subnet:

192.168.50.0/24(the new gateway address was 192.168.50.2).

To avoid IP address conflicts with the DHCP range, static IPs were manually assigned outside the default dynamic allocation pool:

Attacker VM (Kali): 192.168.50.10

Snort IDS VM (Kali-Snort): 192.168.50.20

This provided direct communication between the two systems while isolating them from the external internet. The static IP addresses also made it easier to write and test custom Snort rules since traffic from the attacker's IP could be identified and tracked quickly.

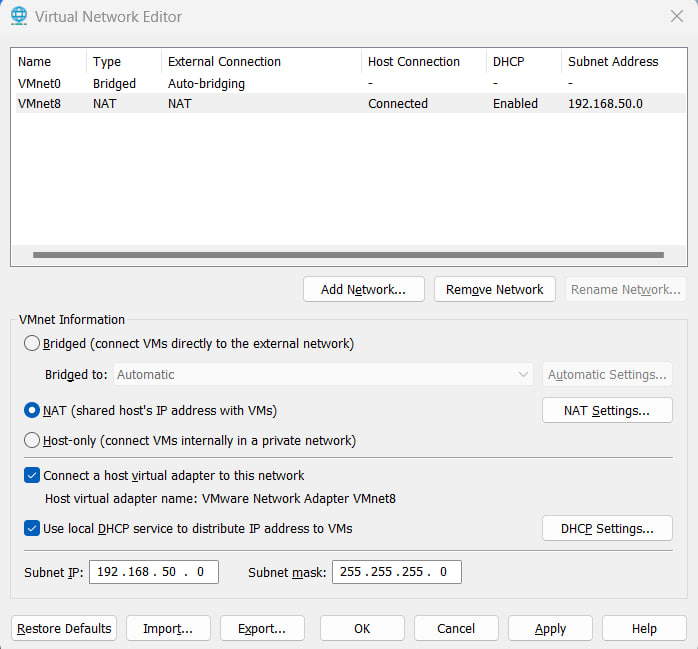


Figure 5: VMnet8 NAT Network Configuration

This shows the configuration of VMnet8 in NAT mode with a custom subnet 192.168.50.0. DHCP is enabled. Both Kali machines are connected to this network via Custom → VMnet8 for isolated communication.



Figure 6: NAT Settings for VMnet8

Figure 6 displays the NAT gateway IP 192.168.50.2, used as the default gateway in static IP settings. No port forwarding is required, as both VMs communicate internally.

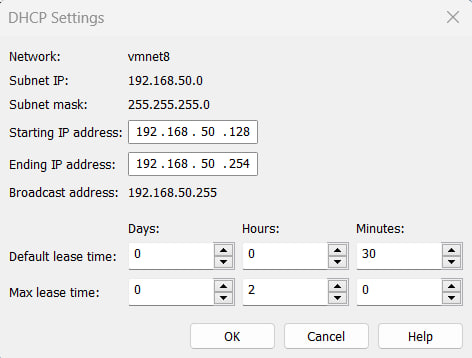


Figure 7: DHCP Settings for VMnet8

This screenshot shows the DHCP range (192.168.50.128–254). Static IPs used for this project (192.168.50.10 and 192.168.50.20) are outside this range to avoid conflicts.

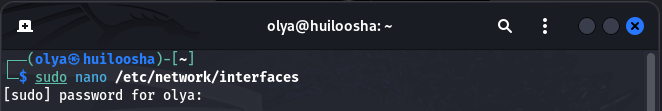


Figure 8: Configuring Network on Kali Attacker

This figure shows the Kali Attacker machine opening the /etc/network/interfaces file to manually configure a static IP address for the network adapter eth0.

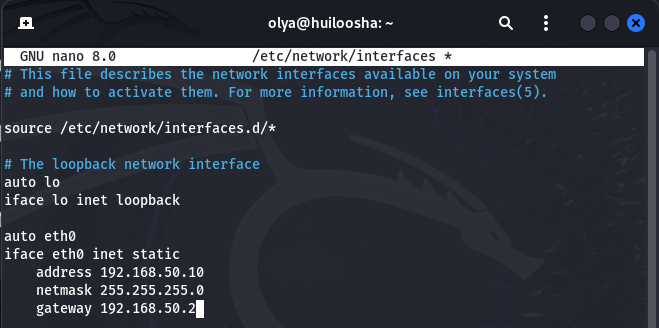


Figure 9: Static IP for Kali Attacker

The interface is configured with the static IP 192.168.50.10, subnet mask 255.255.255.0, and gateway 192.168.50.2 to align with the custom NAT network settings (VMnet8).

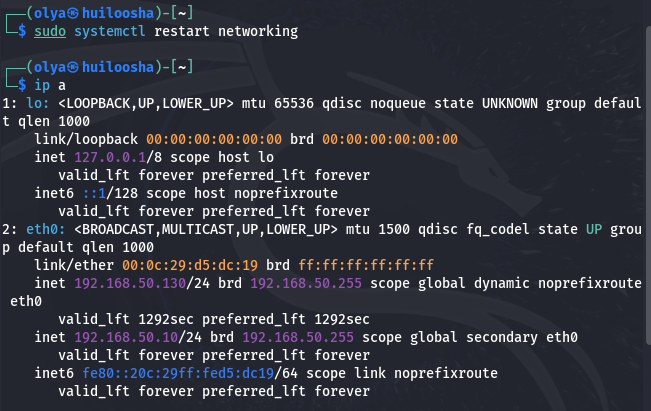


Figure 10: Restart the Network and Check IP

After restarting the networking service, the ip a command confirms that eth0 has been assigned the correct static IP 192.168.50.10, and the network interface is active.

## 5.2 Creating the Second Kali VM via Cloning

In order to emulate an attacker and defender in the same virtual network, two Kali Linux virtual machines were necessary. Rather than installing the second VM from the beginning, the current Kali installation was cloned within VMware Workstation Pro to save time and maintain consistency.

A full clone was created through "Manage → Clone" in VMware. The cloning process created a full copy of the original VM that could operate as a separate VM.

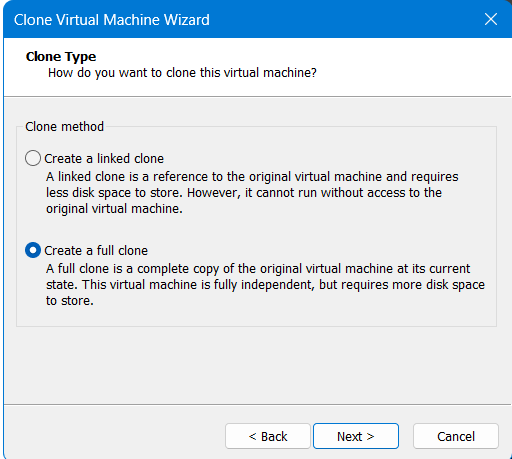


Figure 11: Cloning Kali Linux

After cloning, several configuration steps were completed to avoid conflicts:

* The hostname was changed to kali-snort using the command below:

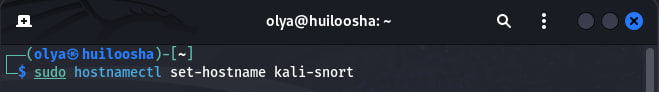


Figure 12: Setting the Hostname on Kali Clone VM

* Updating the /etc/hosts accordingly



Figure 13: /etc/hosts command

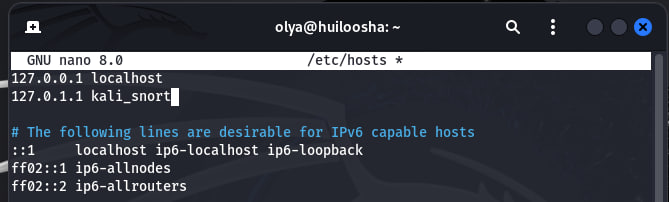


Figure 14: /etc/hosts update

Ctrl + O, Enter → Ctrl + X to save configurations that was made and move on to:

* Reboot the system



Figure 15: Reboot

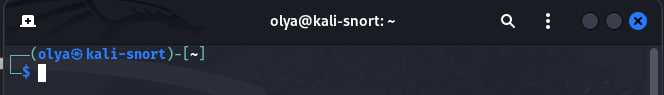


Figure 16: Kali Snort VM

* System identifiers were reset to ensure unique identity:



Figure 17: System ID reset

* Setting up the network on Kali Snort using command below:

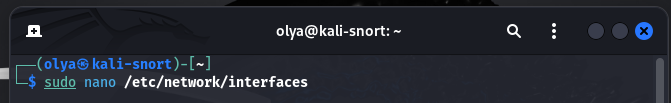


Figure 18: Configuring Network on Kali Snort Command

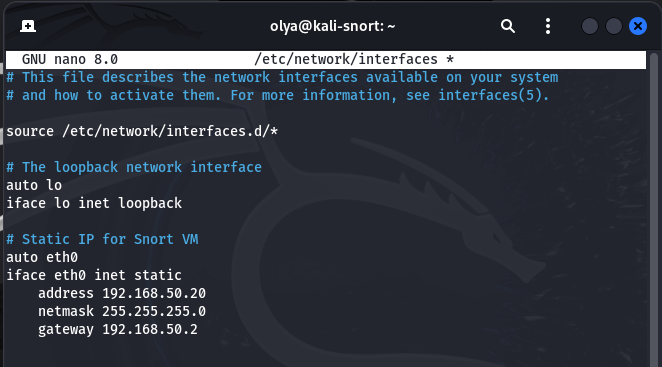


Figure 19: Static IP for Kali Snort

* Restarting the network and checking for correct IP implementation

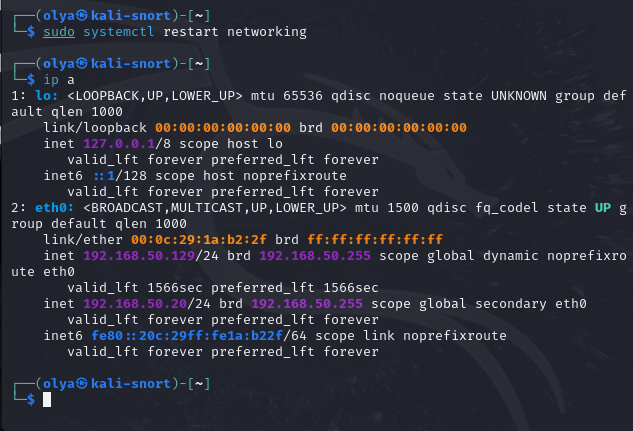


Figure 20: IP Check on Kali Snort

* Testing Ping from Kali Attacker to Kali Snort to ensure connectivity

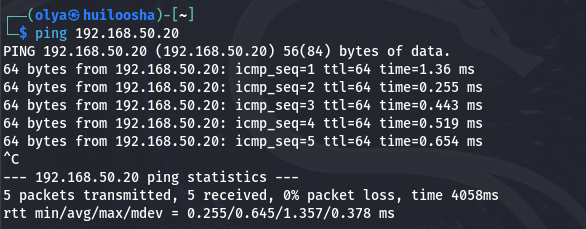


Figure 21: Ping Snort on Attacker

The Attacker VM successfully sends ICMP packets to the Snort VM (192.168.50.20), confirming that both machines are on the same NAT network and can communicate without packet loss.

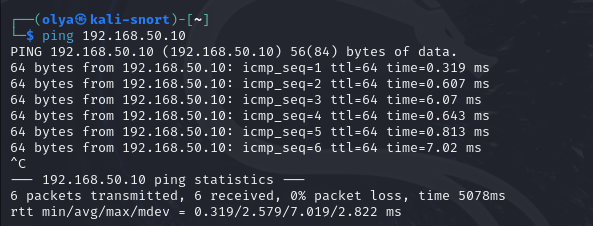


Figure 22: Ping Attacker on Snort

The Snort VM sends ping packets to the Attacker (192.168.50.10) with 0% packet loss, validating full bidirectional connectivity between the two VMs.

## 5.3 Installing Snort

To monitor for potential malicious activity on the network, an open source Intrusion Detection System (IDS) called Snort was installed and configured on the Kali Snort virtual machine. The main goal was to monitor for, alert on, and track suspicious traffic, such as port scans and brute force attacks.

The following commands were executed in the terminal on the kali-snort VM to install and set up Snort:

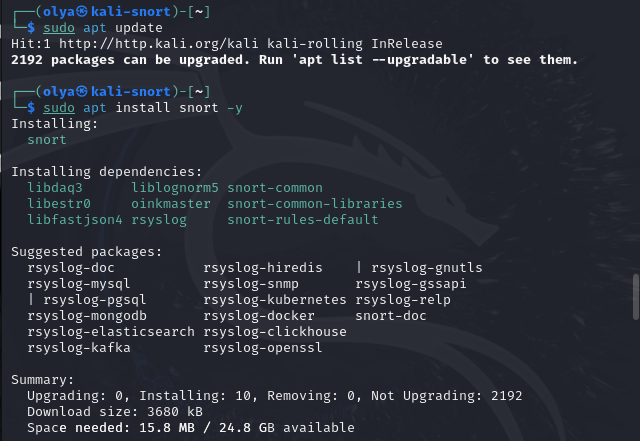


Figure 23: Installation of Snort

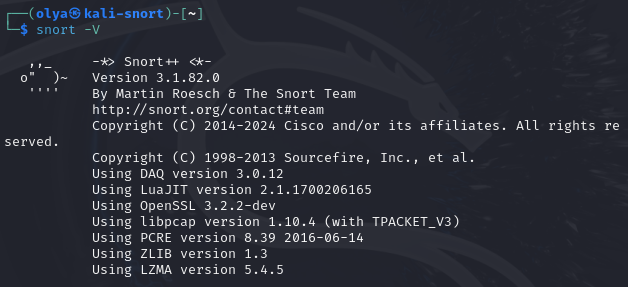


Figure 24: Verifying version

## 5.4 Creating and Testing Snort Rules

As part of the testing phase, a custom Snort rule was created to detect ICMP echo request (ping) traffic, which simulates a basic reconnaissance attempt. This was essential to verify Snort’s ability to capture and alert on suspicious network activity.

To prepare the environment for Snort rule creation and alert logging, the following commands were executed:

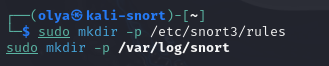


Figure 25: Creating Snort Directories

These commands ensure that:

/etc/snort3/rules — is the custom directory used to store user-defined Snort rules like local.rules

/var/log/snort — is the directory where Snort can store alert logs (if logging mode is used)

After that, the rule file was created and edited with:



Figure 26: Creating Rules File

This file contains the custom ICMP detection rule that was tested in the lab scenario shown below.



Figure 27: Rule scenario

Explanation:

* alert: type of action
* icmp: protocol
* any any -> any any: source/destination IP and port
* itype:8: ICMP Echo Request (ping)
* msg: alert message
* sid: Snort ID for the rule
* rev: rule revision number

Snort was launched manually using the following command (due to broken snort.lua config):



Figure 28: Running Snort with the Rule

This method allowed bypassing Lua configuration issues while still using the ruleset.

Generating ICMP Traffic – from the Attacker VM (192.168.50.10), ping commands were executed:

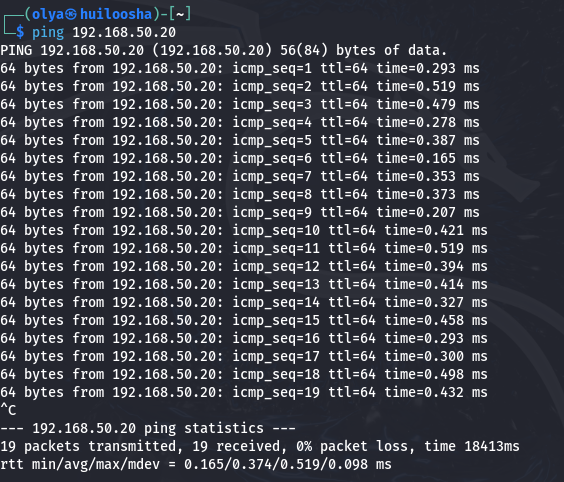


Figure 29: Ping from Attacker

In result below each ping was successfully detected and triggered an alert. Snort displayed total packet counts, alerts, and protocol breakdown:

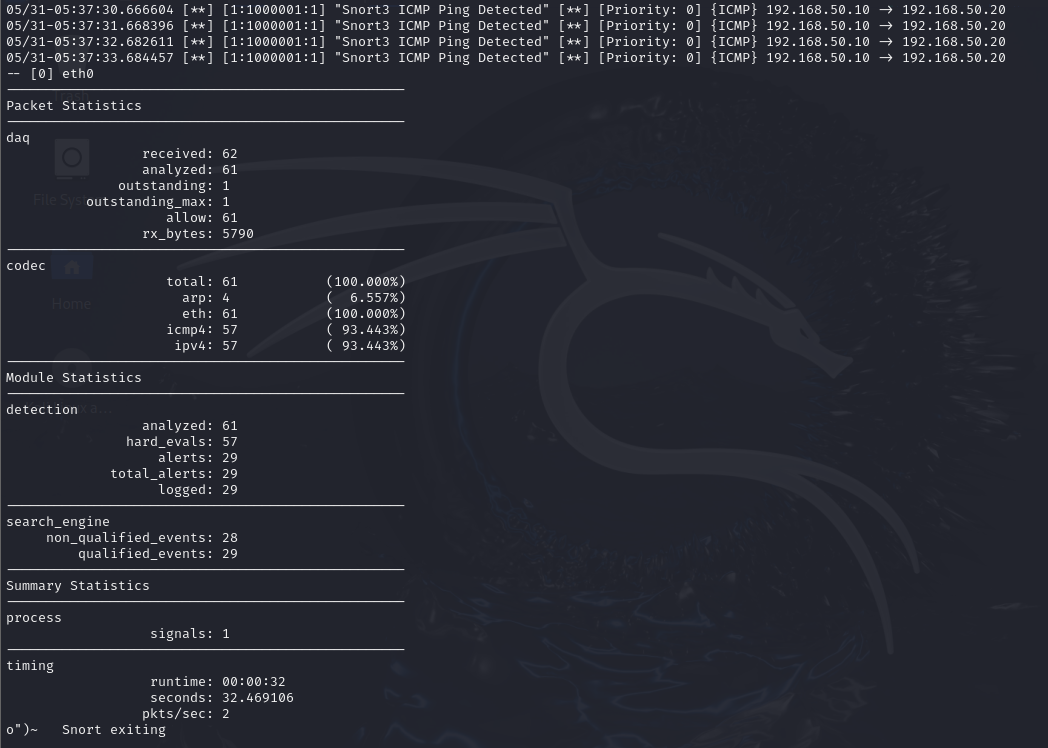


Figure 30: Result of Snort Alert

The ICMP detection rule worked as expected. Despite initial configuration issues, Snort was successfully run using a simplified setup, and custom rules triggered alerts reliably in a controlled environment.

## 5.5 Simulating Port Scan with Nmap

To evaluate Snort’s capability to detect reconnaissance attempts, a TCP port scan was simulated using the Nmap utility from the attacker machine (192.168.50.10) targeting the Snort machine (192.168.50.20).

1. Open local.rules on Kali Snort machine



Figure 31: Configuring Rules for Snort

1. Creating a Rule to Detect TCP Port Scans

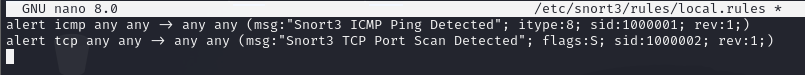


Figure 32: Second Rule Insertion

Explanation:

* Detects TCP packets with the SYN flag (flags:S)
* Can trigger on common Nmap scans like -sS (SYN scan)
* sid:1000002 – unique rule ID

1. Launching Snort with Custom Rule



Figure 33: Launching Snort

Snort was run using the same CLI configuration. This ensured that Snort would actively monitor interface eth0 and alert in real time via the terminal.

1. Running the Port Scan with Nmap



Figure 34: Nmap Attack

On the attacker machine (192.168.50.10), the following Nmap command was used to scan the Snort VM.

1. Result of Test

Snort successfully detected multiple SYN packets and triggered the custom rule:

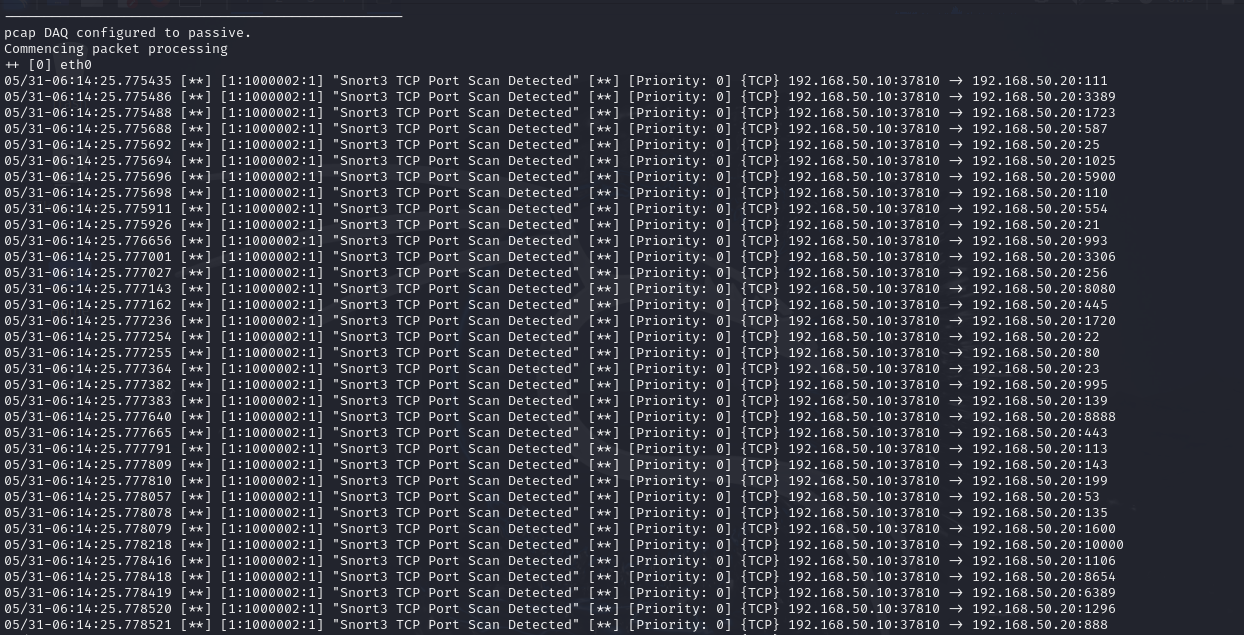


Figure 35: Snort Detection of Nmap Attack

This test confirmed that Snort can detect and log active port scanning attempts. The use of a basic TCP SYN rule was effective in identifying suspicious behavior associated with early attack phases.

## 5.6 Simulating Brute Force with Hydra or Metasploit

To simulate brute force attacks on the network and evaluate Snort’s detection capabilities, the Hydra tool was used to attempt SSH login attacks from the attacker machine (192.168.50.10) toward the Snort machine (192.168.50.20).

Hydra is lightweight, fast, and provides a clear output, making it suitable for detection rule testing.

1. Creating the Snort Rule for SSH Brute Force

Open local.rules again:



Figure 36: local.rules

Adding the following rule to trigger on any TCP traffic to port 22 (SSH) — including brute force attempts:

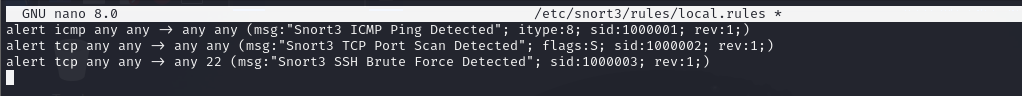


Figure 37: Adding the Rule to trigger brute force attacks

1. Restart Snort

Running Snort again with updated ruleset:



Figure 38: Snort

1. Simulate Brute Force with Hydra

On the attacker machine (192.168.50.10) installing Hydra:

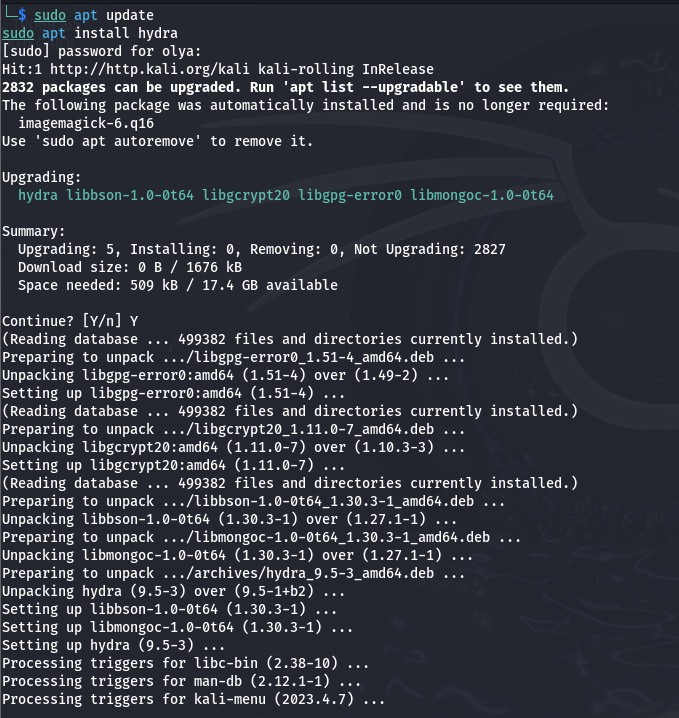


Figure 39: Hydra installation

Then, running the following command to perform an SSH brute force simulation:

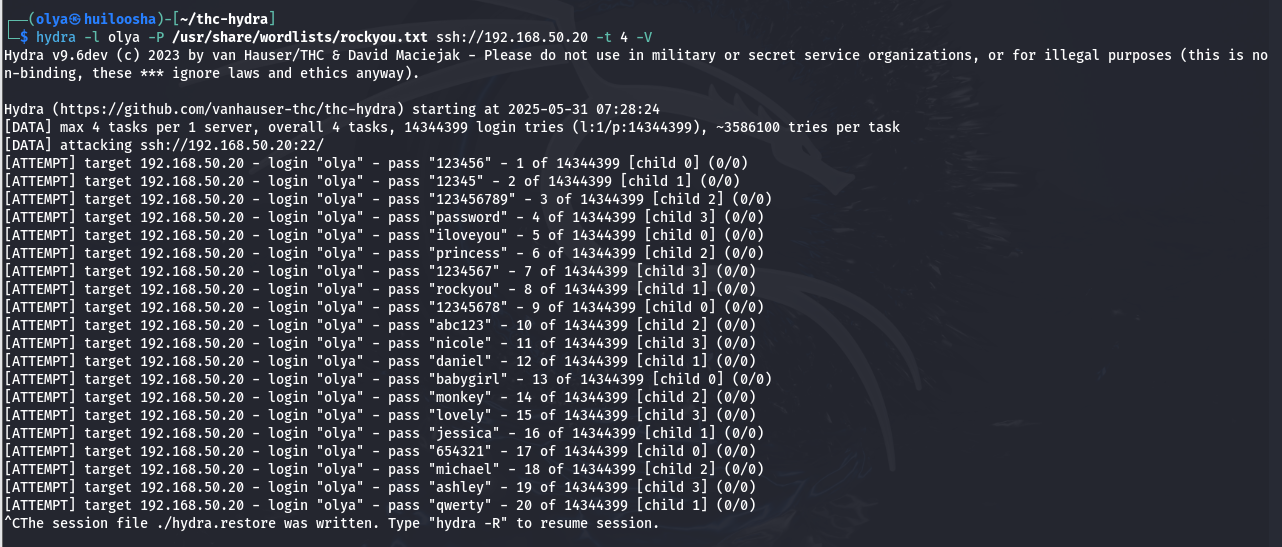


Figure 40: Attack with Hydra

Explanation:

* -l olya specifies the username to target
* -P points to a password wordlist (e.g., rockyou.txt)
* ssh://192.168.50.20 is the IP and protocol of the Snort machine

This simulates a real brute-force login attack using multiple passwords.

1. Snort Detects It

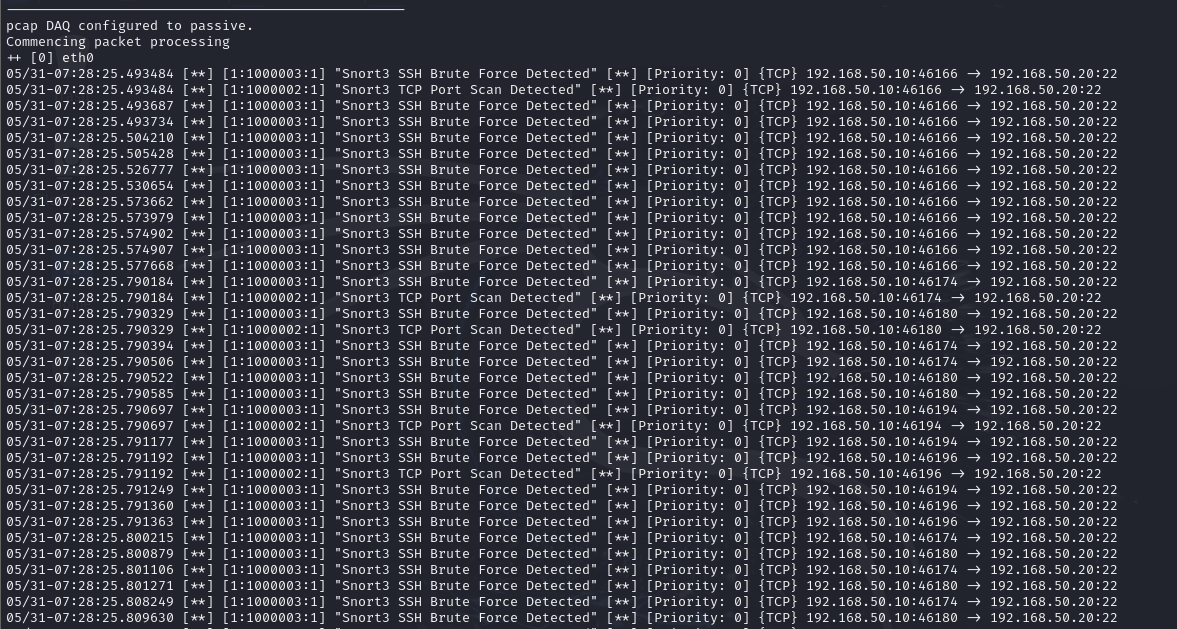


Figure 41: Snort Detection

While Hydra is running, Snort should trigger the rule added earlier and display real-time alerts in the terminal like:

In addition, Snort's summary statistics will reflect the number of TCP packets received and the triggered rule count. This confirms that Snort is actively monitoring port 22 and successfully detecting unauthorized access attempts.

## 5.7 Capturing and Analyzing Logs

### 5.7.1 ICMP Ping Simulation – Log Analysis

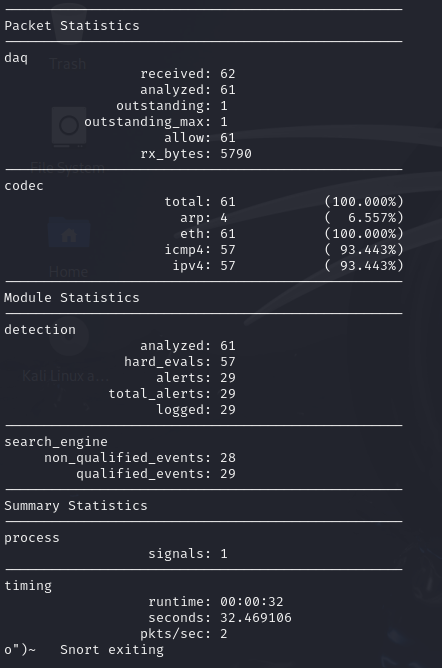


Figure 42: ISMP Analysis

In the ICMP simulation (ping from 192.168.50.10 to 192.168.50.20), Snort successfully detected and logged the attack:

* Packets received: 62
* Analyzed: 61
* Alerts triggered: 29
* ICMP traffic share: ~93%
* Logged alerts: 29

This confirms that the rule designed to detect ICMP traffic (alert icmp any any -> any any) was correctly configured and triggered multiple times, demonstrating real-time detection of ping-based reconnaissance.

### 5.7.2 Nmap Port Scan Simulation – Log Analysis

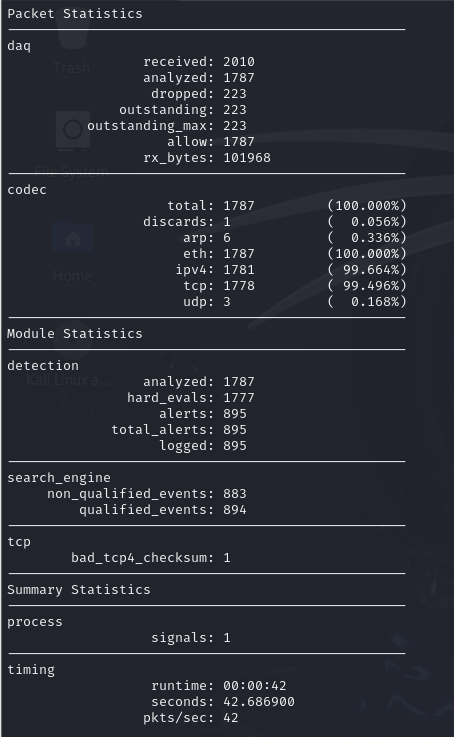


Figure 43: TCP Analysis

During the simulated port scan with Nmap, Snort recorded:

* Packets received: 2010
* Analyzed: 1787
* Alerts triggered: 895
* IPv4 packets: 1781
* TCP traffic: ~99%
* bad\_tcp4\_checksum events: 1

The large volume of TCP packets, combined with 895 triggered alerts, shows that Snort effectively detected the abnormal traffic pattern generated by the Nmap scanner. The rule was sensitive enough to identify multiple scan attempts across ports.

### 5.7.3 Hydra Brute Force Simulation – Log Analysis

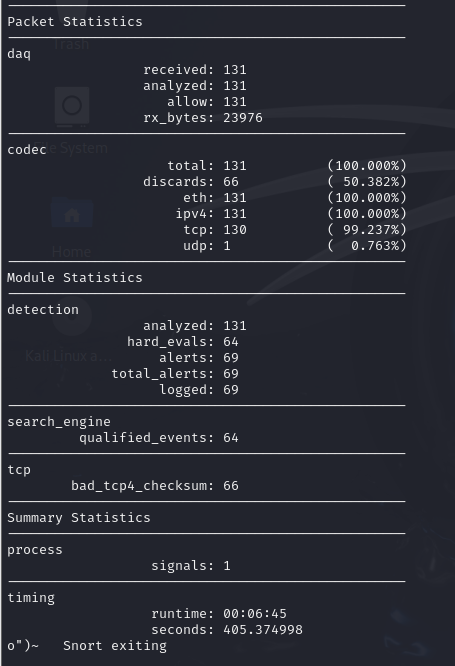


Figure 44: SSH Analysis

When executing Hydra against the Snort machine’s SSH service:

* Packets received: 131
* Analyzed: 131
* Alerts triggered: 69
* TCP traffic: ~99%
* Checksum errors: 66 (bad\_tcp4\_checksum)
* Logged alerts: 69

The system generated a high number of alerts, indicating that the rule for SSH login attempts was activated. The presence of malformed packets further supports that the traffic was not standard and originated from automated brute-force software.

To summarize, all three types of simulated attacks were successfully detected by Snort++ using custom rules:

* ICMP Ping Attack: 29 alerts triggered for standard ping packets, showing correct detection of reconnaissance attempts.
* Nmap Port Scanning: 895 alerts generated, confirming that Snort accurately identified abnormal port scanning behavior with a high number of TCP packets.
* SSH Brute-Force with Hydra: 69 alerts logged, indicating effective detection of multiple SSH login attempts. Anomalies such as bad TCP checksums were also recorded.

Snort’s internal statistics confirmed reliable real-time packet analysis, rule matching, and alert generation across various network attack types, demonstrating its efficiency as an intrusion detection system (IDS).

# 6.0 Testing and Evaluation

## 6.1 Detection Accuracy

Snort++ accurately recognized or logged all simulated attacks:

* ICMP ping packets traffic between the attacker machine and Snort machine generated 29 alerts.
* Nmap port scanning produced 895 alerts and confirmed known aggressive port access behavior.
* Hydra brute-force had 69 alerts from repeated SSH logins.

All rules are locally defined in local.rules. The action of all these rules indicates that Snort was able to effectively match traffic patterns to the defined conditions, establishing accuracy of detection when applied to common reconnaissance and intrusion methods.

## 6.2 False Positives / Limitations

The simulations did not reveal any false alerts because all alerts were anticipated alerts with real attacks. However, there were some limitations:

* The alert\_fast.txt log file was not always reported as it should have been, and analysis of some would need to be manual on the basis of snort's runtime statistics.
* Some issues arose from the triggering of snort.lua modules that were either misconfigured or did not exist (e.g., detection.rules, daq.interface) that required some tweaking for smooth batch execution.
* SSH bruteforcing attempts with Hydra first failed because the target machine had SSH disabled, which highlights the need for appropriate environmental preparation.

These limitations demonstrate the importance of having the proper configuration settings, complete rule paths, and running services to generate noticeable results.

## 6.3 Overall Effectiveness

While there were challenges in setting up Snort++ on the Virtual Machine (VM), it was a valid IDS and effective in identifying varying packets attacks in different test run environments:

* The software handled packets efficiently (eg. 42 pkts/sec in ICMP test).
* The software logged the alerts, capturing the intended alerts as configured in the rule definitions.
* The packet decoding, determined evaluation, has an accurate logging across all simulations.

All in all, Snort++ indicated high confidence at identifying attacks most effectively in real-time, exhibited strong module capability, detailed logged packet statistics and alert notification behavior when it was configured correctly.

# 7.0 Conclusion

This project demonstrated how to use Snort++ in a virtual network as an Intrusion Detection System (IDS). Two Kali Linux machines were setup in VMware, one as an attacker and the other running Snort.

After setting the network and Snort rules, we tested ICMP ping, Nmap scans, and Hydra brute-force attacks. We had a few setup problems, but were able to detect and log every type of attack successfully.

Overall, this project gave me hands-on experience with Snort++, and emphasized the significance of configuring systems properly in network security.

# 8.0 Reflection

Working on this Snort IDS project was by no means easy but it was all very educational. I started off having problems with the network I was trying to create out of two separate Kali Linux virtual machines in VMware. Initially, I had many issues with IP configuration, due to problems with cloning VMs and the pills of connectivity. Ultimately what I ended up doing was research, trial and error, and constructing a working and stable NAT network with VMnet8, and giving static IP addresses to both the attacker and the Snort machines.

Installing and configuring Snort 3 was also not as easy as I expected. I had errors related to the snort.lua configuration, and issues of missing DAQ interfaces. The process of fixing these issues really did provide a better understanding of how Snort loads its components, though, and how the rule files were managed. Once I started to create my own custom detection rules, I began to learn more about packet structures and what kinds of patterns could indicate an attack.

Actually testing the system and ensuring everything was working as I expected using ICMP, Nmap and Hydra simulations provided me with a practical understanding of network threats. During the analysis of the logs, I could see the way Snort captured critical information and the way alerts would assist with threat-related analysis.

Overall, I think this project provided me with an opportunity to enhance my practical skills in intrusion detection, Linux configuration, and security-based analysis. It was my first genuine experience with troubleshooting real world events and constructing a working IDS from the base up.

# 9.0 References

Chandran, A. (2023, September 17). Ethical Hacking: 5 Phases, Techniques, and Tools - Ajith Chandran - Medium. *Medium*. <https://medium.com/@ajithchandranr/ethical-hacking-5-phases-techniques-and-tools-e760d092f70e>

Tokyoneon. (2018, April 17). *How to Automate Brute-Force attacks for NMAP scans*. Null Byte. <https://null-byte.wonderhowto.com/how-to/automate-brute-force-attacks-for-nmap-scans-0184132/>

Borikar, K. (2024, December 24). SnOrt IDS Lab: Real-Time intrusion detection in a vulnerable network. *Medium*. <https://medium.com/@borikarkush/snort-ids-lab-real-time-intrusion-detection-in-a-vulnerable-network-b0927c675c39>

Layne, J. (2025, March 21). Types of NMAP scans explained | Luxwisp. *Ablison*. <https://www.luxwisp.com/types-of-nmap-scans-explained/>

Mitchell, A. (2024, August 28). *How to use Hydra to hack passwords - Penetration testing tutorial - ExpertBeacon*. Expertbeacon. <https://expertbeacon.com/how-to-use-hydra-to-hack-passwords-penetration-testing-tutorial/>

*Network Security: Analyzing Metasploit and Hydra tools*. (n.d.). <https://desklib.com/study-documents/metasploit-hydra-comparison/>