

# Deployment of Intrusion Detection System (IDS):Snort

Supervised by :Mr.Triki Bayrem

Prepared by : Mahmoud Ben Hloua

# Academic Year 2024/2025

## 1-Introduction:

The rising complexity of cyber threats demands robust security measures to protect sensitive information and critical infrastructure. Intrusion Detection Systems (IDS) have become essential tools for monitoring and analyzing network traffic for malicious activity. Among these, Snort stands out as a widely-used, open-source solution.

Snort, a Intrusion Detection and Prevention System (IDPS), supports real-time traffic analysis and packet logging. It can function as:

- **Network Intrusion Detection System (NIDS)** : Monitors network traffic for threats.
- **Host Intrusion Detection System (HIDS)**: Monitors a specific host or device.
- **Hybrid IDS/IPS**: Combines detection with prevention by blocking threats.

This project focuses on deploying Snort as a Network Intrusion Detection System (NIDS) to detect specific attacks by analyzing network packets against a database of known attack patterns. The goal is to examine its deployment, configuration, and optimization to strengthen network security while highlighting its functionality as a NIDS.



## 2- State of the Art :

Intrusion Detection Systems (IDS) have evolved significantly over the decades. Initially, in the **1970s–1980s**, they relied on manual log audits to detect suspicious activities. By the **1990s**, automated systems emerged, leveraging signature-based detection methods to identify known attack patterns.

**Snort**, released in 1998 by Martin Roesch, became a pioneer in open-source IDS. It gained popularity for its simplicity, flexibility, and real-time traffic analysis. Over the years, it evolved into a powerful Intrusion Detection and Prevention System (IDPS), capable of acting as a Network Intrusion Detection System (NIDS), Host Intrusion Detection System (HIDS), or a Hybrid IDS/IPS.

Today, Snort remains one of the most widely deployed IDS solutions, competing with alternatives like **Suricata** (known for multi-threading) and **Bro/Zeek** (focused on high-level traffic analysis), while proprietary systems like **Cisco Secure IDS** offer integrated enterprise solutions.

This historical evolution underscores Snort's relevance in addressing modern network security challenges.



## 3-Existing Solutions:

Intrusion Detection Systems (IDS) can be categorized into open-source and proprietary solutions, each with its strengths and applications.

- Open-Source Solutions :
  - ♦ **Snort**
  - ♦ **Suricata**
  - ♦ Zeek (formerly Bro)
- Proprietary Solutions :
  - ♦ **Cisco Secure IDS**
  - ♦ **Palo Alto Networks**
  - ♦ **McAfee Network Security Platform**

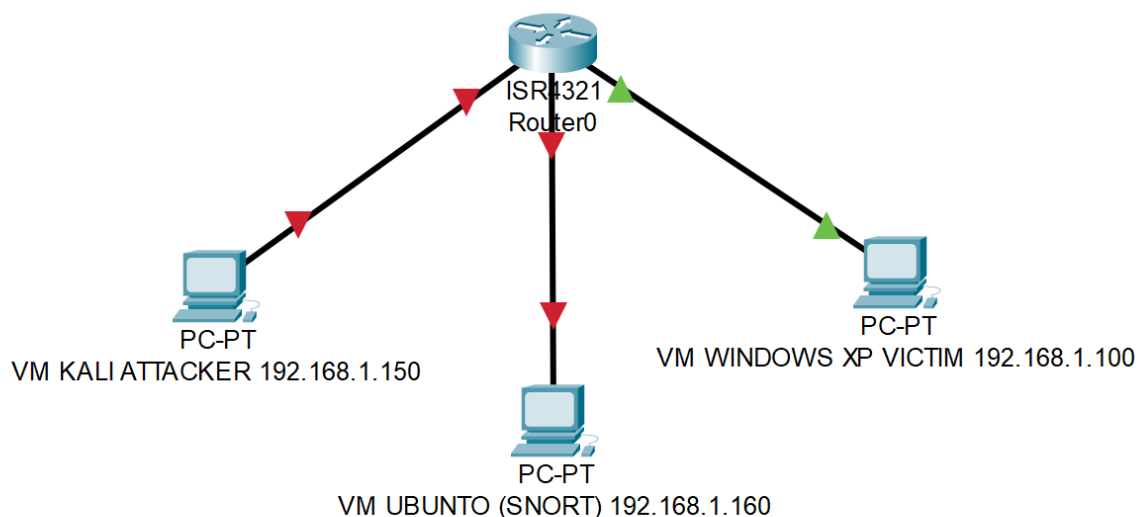


## 4- Comparative of Existing Solution :

For this project, which focuses on detecting specific attacks such as **DHCP starvation**, **ARP poisoning**, and **SYN flood**, Snort is the most suitable choice due to its precision and adaptability. Its **signature-based detection** mechanism allows for precise identification of known attack patterns, making it highly effective for the targeted threats. Additionally, Snort's ability to perform **real-time traffic analysis** ensures timely detection and alerts, which is critical for fast-evolving attacks like ARP poisoning. Its **modular and flexible design** enables customization to meet specific security

requirements, while its **open-source nature** makes it a cost-effective solution compared to proprietary IDS. Furthermore, Snort benefits from an active community that provides regular updates, rule enhancements, and troubleshooting support, ensuring it remains robust and reliable. While alternatives like Suricata excel in multi-threaded environments and Zeek is better for high-level traffic analysis, Snort's comprehensive feature set and strong community backing make it the most practical and efficient choice for this project.

## 5-Topology :



## 5- Scenario :

In this project, the deployment of Snort as a Network Intrusion Detection System (NIDS) is tailored to detect and analyze specific attack scenarios within a controlled network environment. These scenarios simulate real-world threats to evaluate Snort's effectiveness in monitoring and securing network traffic.

★ Scenario 1 : DHCP Starvation Attack :

- **Description** : An attacker floods the network's DHCP server with fake requests, exhausting available IP addresses and denying legitimate devices access to the network.
  - **Objective** : Use Snort to detect the excessive DHCP requests and alert the administrator.
- ★ Scenario 2 : ARP Poisoning Attack :
- **Description**: Malicious actors spoof ARP messages to redirect network traffic through their device, enabling man-in-the-middle attacks.
  - **Objective**: Configure Snort to identify ARP anomalies and flag suspicious MAC-IP associations.
- ★ Scenario 3 : Nmap Scanning :
- **Description**: Attackers use Nmap to scan the network for open ports and vulnerabilities to exploit.
  - **Objective**: Deploy Snort rules to detect and log port-scanning activities, categorizing them as reconnaissance threats.
- Each scenario represents a critical aspect of network security, allowing the project to demonstrate how Snort can be configured and optimized to detect these threats effectively, ensuring timely response and enhanced network protection.

## 6- Downloading and Configuring Snort :

### 1. Installing Snort 2.9 on Ubuntu :

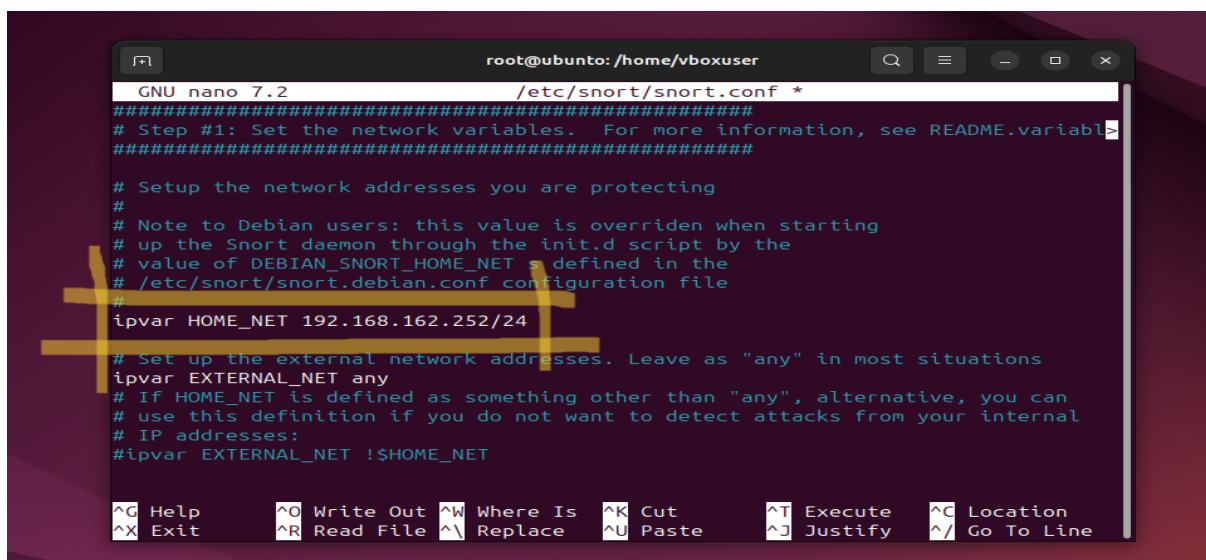
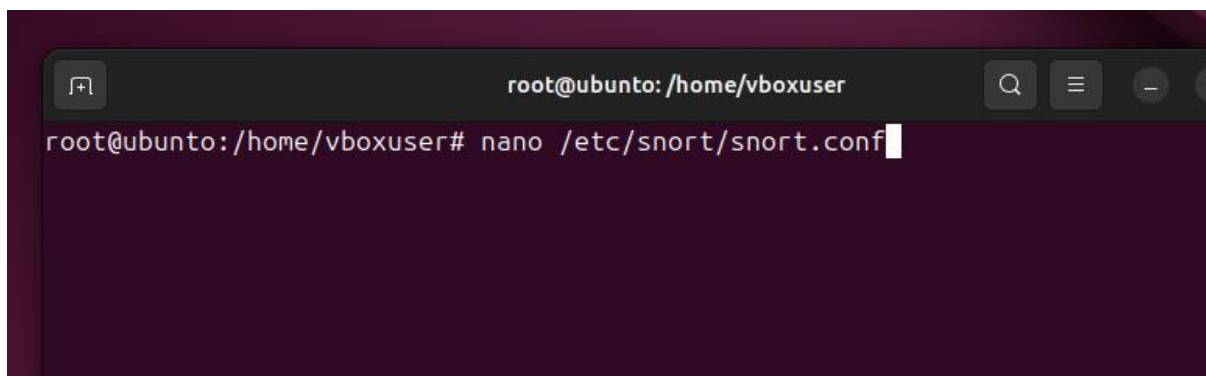
```
:-$ sudo apt install snort
```

### 2. Verify the installation :

```
:-$ snort --version
```

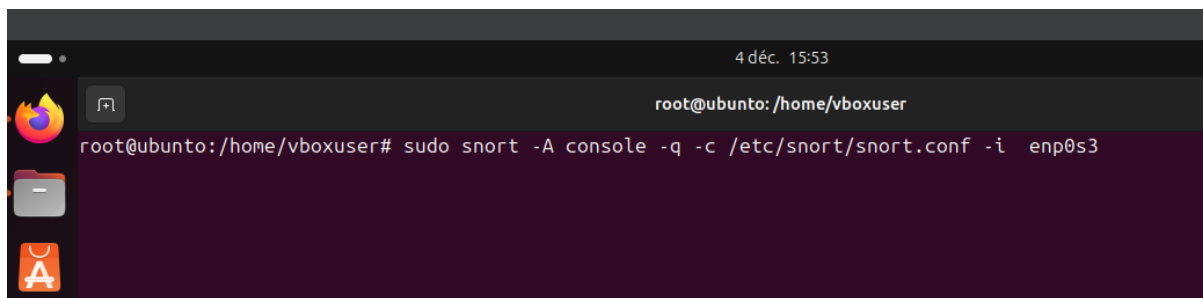
```
.,_      -*> Snort! <*-
o" )~    Version 2.9.15.1 GRE (Build 15125)
' ''     By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
          Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
          Copyright (C) 1998-2013 Sourcefire, Inc., et al.
          Using libpcap version 1.10.1 (with TPACKET_V3)
          Using PCRE version: 8.39 2016-06-14
          Using ZLIB version: 1.2.11
```

3. Open the `/etc/snort/snort.conf` file and configure the `HOME_NET` variable with your local IP address or network range.



## 7- Scenario 1 : DHCP Starvation :

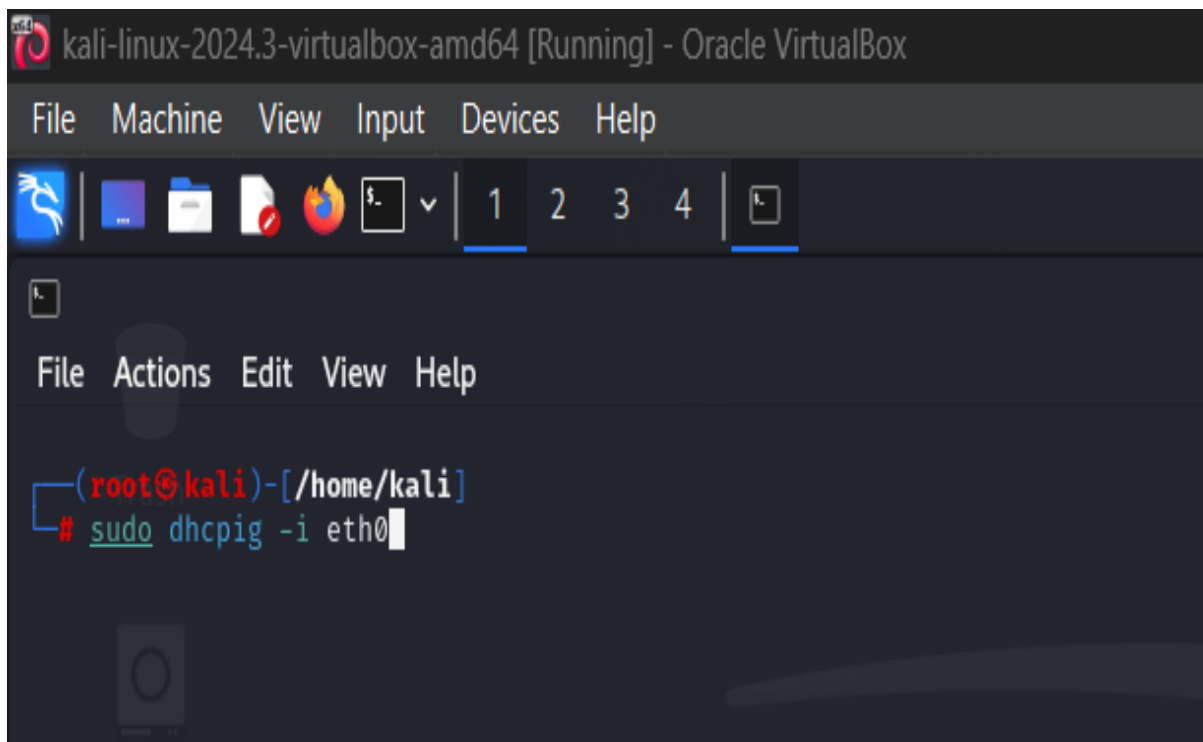
I will configure Snort to listen to network traffic using the following command

A terminal window on a Linux system. The title bar shows the date and time as '4 déc. 15:53'. The prompt is 'root@ubuntu:/home/vboxuser'. The command entered is 'sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3'.

```
root@ubuntu:/home/vboxuser# sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
```

To simulate a DHCP starvation attack, I will use the **DHCPig** tool. This attack floods the DHCP server with a high number of DHCP requests, attempting to exhaust the pool of available IP addresses. This test will evaluate Snort's ability to detect and alert on such malicious activity.

using the following command :

A screenshot of a Kali Linux virtual machine running in Oracle VM VirtualBox. The window title is 'kali-linux-2024.3-virtualbox-amd64 [Running] - Oracle VirtualBox'. The terminal shows the prompt '(root@kali)-[/home/kali]' and the command '# sudo dhcpig -i eth0' being entered.

```
(root@kali)-[/home/kali]  
# sudo dhcpig -i eth0
```

After launching the DHCP starvation attack, we can observe the following results :



```
[-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.90 for MAC=[de:ad:14:06:69:41:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.90
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.177 for MAC=[de:ad:17:44:d7:f3:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.177
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.205 for MAC=[de:ad:0e:7e:66:40:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.205
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.64 for MAC=[de:ad:0e:1e:88:01:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.64
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.212 for MAC=[de:ad:23:08:4f:ca:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.212
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.50 for MAC=[de:ad:16:3f:41:0d:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.50
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.228 for MAC=[de:ad:26:23:84:8c:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.228
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.161 for MAC=[de:ad:15:28:52:71:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.161
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.215 for MAC=[de:ad:25:33:13:d5:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.215
DBG ARP_Request 192.168.240.214 from 192.168.240.136
[<-->] ARP_Response 192.168.240.214 : 0a:b0:63:8f:38:ef
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.116 for MAC=[de:ad:1b:66:98:cf:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.116
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.67 for MAC=[de:ad:07:3f:7f:f3:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.67
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.162 for MAC=[de:ad:06:11:00:f9:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.162
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.173 for MAC=[de:ad:27:65:b0:e6:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.173
[<-->] ? waiting for DHCP pool exhaustion...
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.10 for MAC=[de:ad:0c:63:25:e6:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.10
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.183 for MAC=[de:ad:18:43:19:b8:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.183
[<-->] DHCP_Discover
[<-->] DHCP_Offer 0a:b0:63:8f:38:ef 192.168.240.214 IP: 192.168.240.163 for MAC=[de:ad:14:1f:0a:c1:00:00:00:00:00:00:00]
[<-->] DHCP_Request 192.168.240.163
```

Snort Alerts :

```
4 déc. 15:55
root@ubuntu:/home/vboxuser

2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:00.187427 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:00.619378 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:01.055464 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:01.493876 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:01.933620 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:02.358036 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:02.789097 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:03.225995 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:03.657983 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:04.088900 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:04.520871 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:04.952737 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:05.385362 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:05.820117 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
12/04-15:55:06.252449 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority:
2] {UDP} 0.0.0.0:68 -> 255.255.255.255:67
```

## 8- Scenario 2 : ARP Poisoning Attack :

Use Ettercap tool in kali



Open a http site web like <http://vulnweb.com>

## Login

Username

Bayrem.triki

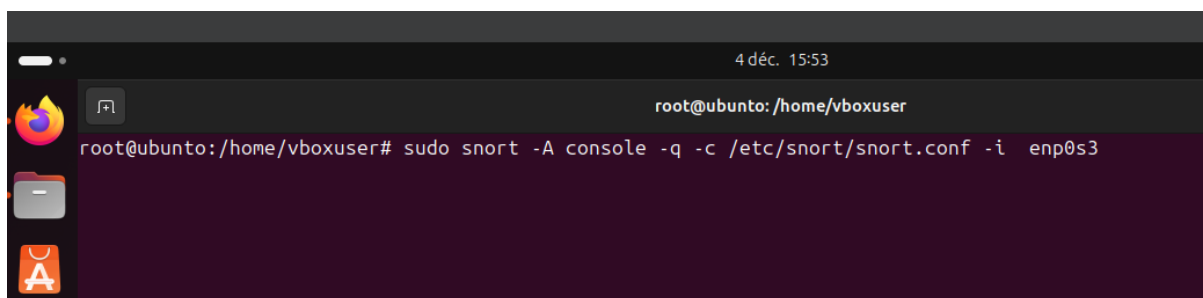
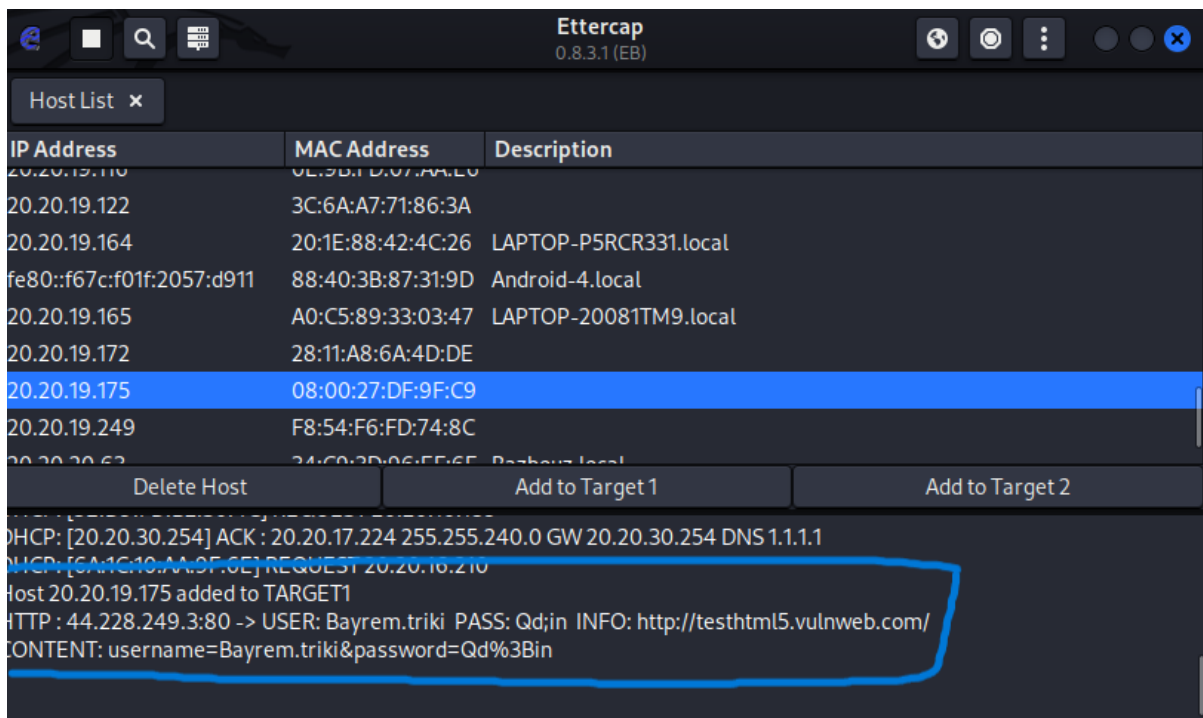
Password

•••••

Forgot Pwd?

Login

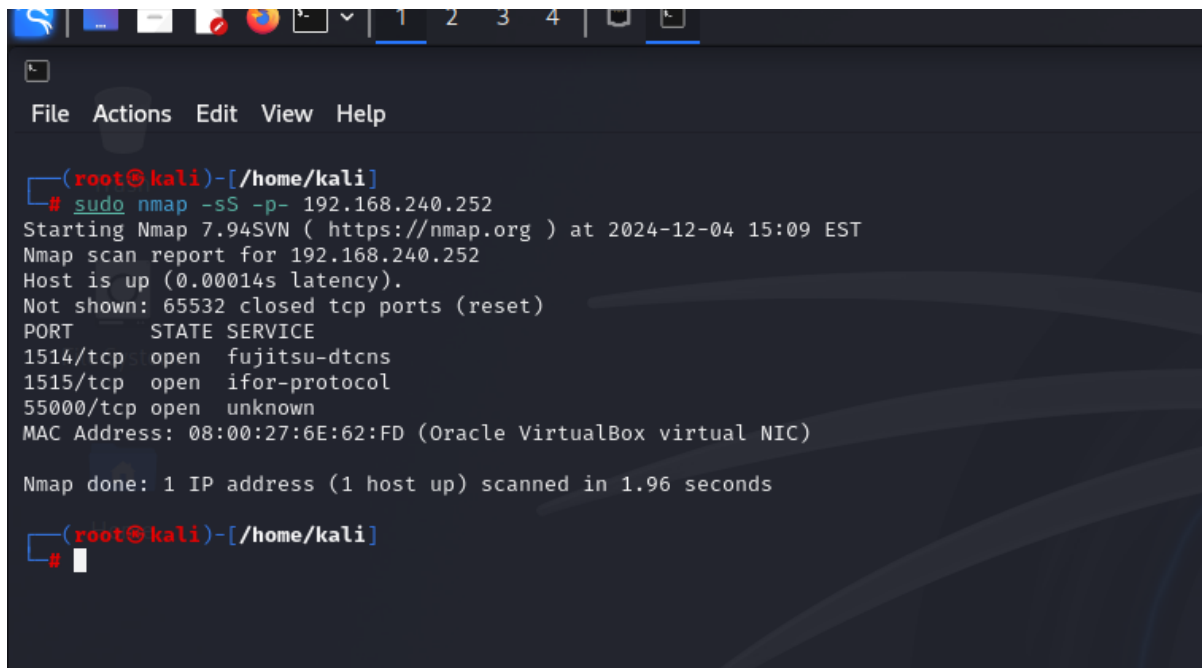
Close



Snort cannot detect ARP poisoning attacks because it operates at Layer 3 (Network Layer) and above, while ARP poisoning is a Layer 2 (Data Link Layer) attack, which is outside Snort's monitoring capabilities

## 9- Scenario 3 : Nmap Scanning :

perform an Nmap scan using Kali Linux and observe the results in the log file

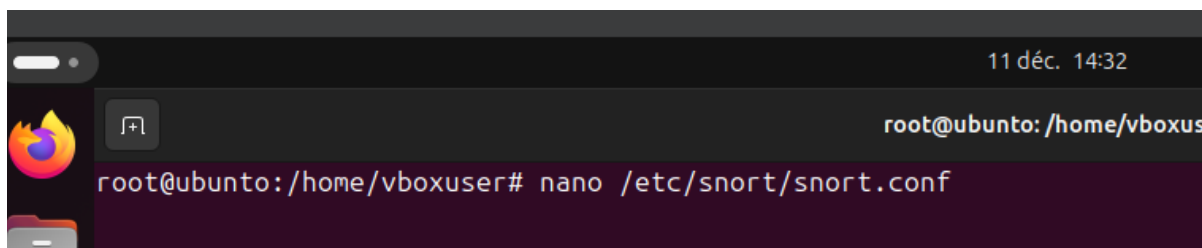
A terminal window with a dark background and light text. The prompt is (root@kali)-[/home/kali]. The user enters # sudo nmap -ss -p- 192.168.240.252. The output shows the Nmap scan report for 192.168.240.252, indicating the host is up and listing open ports 1514/tcp (fujitsu-dtcns), 1515/tcp (ifor-protocol), and 55000/tcp (unknown). The scan is completed in 1.96 seconds.

```
(root@kali)-[/home/kali]
# sudo nmap -ss -p- 192.168.240.252
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 15:09 EST
Nmap scan report for 192.168.240.252
Host is up (0.00014s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE
1514/tcp  open  fujitsu-dtcns
1515/tcp  open  ifor-protocol
55000/tcp open  unknown
MAC Address: 08:00:27:6E:62:FD (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.96 seconds

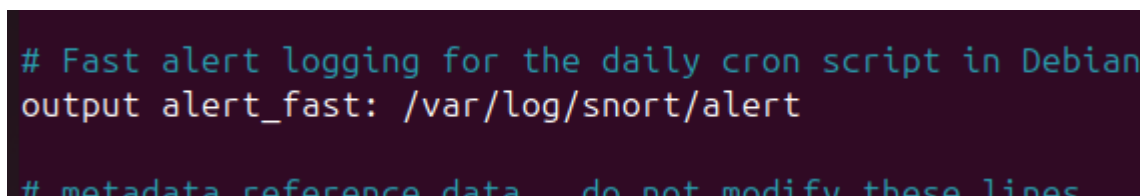
(root@kali)-[/home/kali]
#
```

If Snort does not detect the scan, it may require additional configuration. Open the configuration file at `/etc/snort/snort.conf` to adjust the settings

A terminal window with a dark background and light text. The prompt is root@ubuntu:/home/vboxuser#. The user enters nano /etc/snort/snort.conf. The window title bar shows the date and time as 11 déc. 14:32.

```
11 déc. 14:32
root@ubuntu:/home/vboxuser# nano /etc/snort/snort.conf
```

To view the logs, check the file located at `/var/log/snort/alert`

A terminal window with a dark background and light text. The content shows the first few lines of the /var/log/snort/alert file, including a comment about fast alert logging and a line for output alert\_fast.

```
# Fast alert logging for the daily cron script in Debian
output alert_fast: /var/log/snort/alert

# metadata reference data... do not modify these lines
```

When you run Snort, it will detect the Nmap scan (if properly configured)

```

root@ubuntu: /home/vboxuser
00
11/14-21:37:51.145136  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/14-21:37:51.145262  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/14-21:37:52.175698  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/14-21:37:52.175783  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/14-21:37:53.202566  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/14-21:37:53.202661  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/14-21:37:54.247925  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/14-21:37:54.248012  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
00

```

You can view the logs recorded by Snort by opening the alert file

Open ▾  alert /var/log/snort

wazuh-install.sh alert x  snort.conf alerts.log  local.rules  local.rules local.rules

```

11/21-21:41:26.152260  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/21-21:41:26.152281  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.252 -> 192.168.162.50
11/21-21:41:27.197943  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/21-21:41:27.197965  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.252 -> 192.168.162.50
11/21-21:41:28.233009  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/21-21:41:28.233034  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.252 -> 192.168.162.50
11/21-21:41:29.257382  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/21-21:41:29.257402  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.252 -> 192.168.162.50
11/21-21:41:30.276745  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/21-21:41:30.276781  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.252 -> 192.168.162.50
11/21-21:41:31.301329  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252
11/21-21:41:31.301350  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.252 -> 192.168.162.50
11/21-21:41:32.324539  [**] [1:1000002:1] icmp connection attempt [**] [Priority: 0] {ICMP} 192.168.162.50 -> 192.168.162.252

```

### **what you can do with the file log :**

After generating logs in a file, you have multiple options for utilizing them to enhance security monitoring and response:

- Send Alerts via Email : You can configure a system to parse the logs and trigger alerts based on specific criteria (e.g., suspicious activities, failed login attempts). These alerts can be sent to an email address, allowing administrators to receive real-time notifications and take immediate action.
- Upload Logs to a SIEM Server like Wazuh : Logs can be forwarded to a Security Information and Event Management (SIEM) server such as Wazuh. Wazuh collects, centralizes, and analyzes logs from various sources. By uploading the logs

