Activité 1 – Attaque MITM d'un service SSH et mise en place de contre-mesures

Matériel

- Un serveur Proxmox
- Une VM debian 12
- Un conteneur serveur sous debian 12
- Un conteneur client sous debian 12
- Un conteneur kali linux
- Un conteneur routeur sous debian 12

Étapes

Laboratoire 1

- Mise en place du laboratoire 1

Partie 1 Attaque MITM d'un service SSH

- Nmap
- L'homme du milieu (man in the middle)
- Connexion SSH

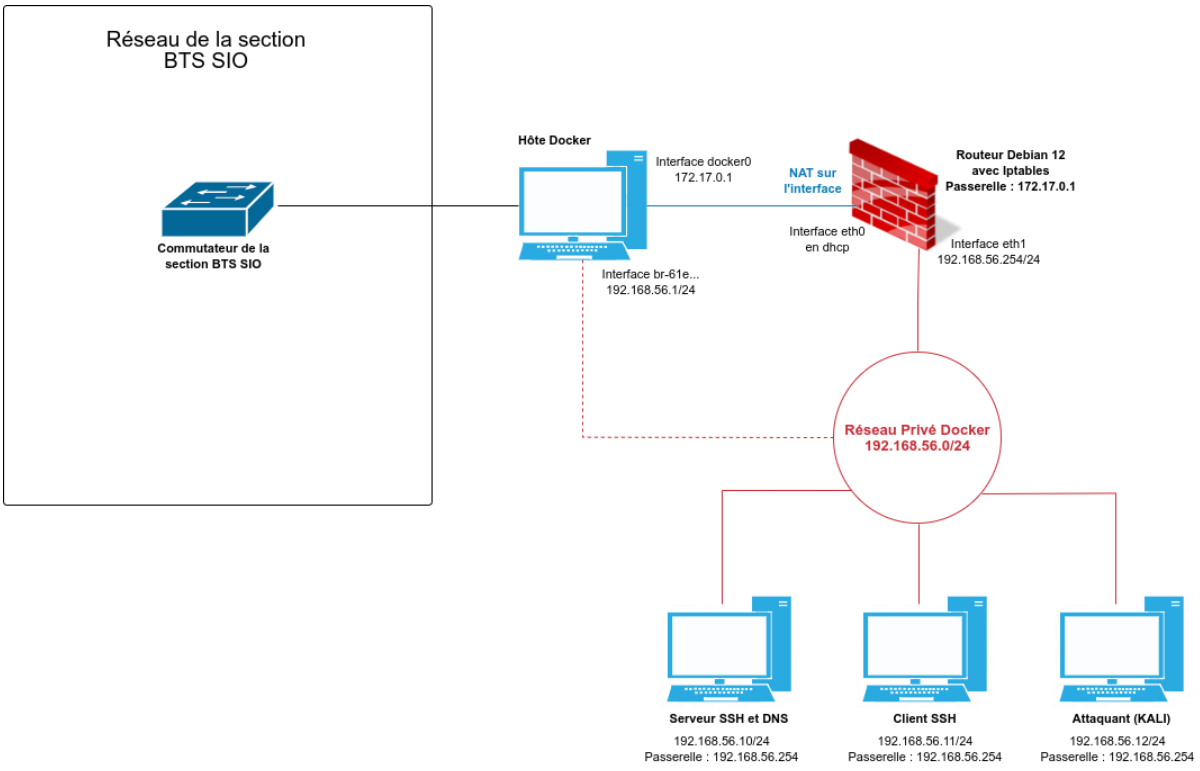
Partie 2 Mise en place de contre-mesures

- Mise en place d'une authentification par clés de chiffrement
- Utilisation de ssh-agent

- Faciliter la vérification de l'identité du serveur SSH avant la première connexion

Partie 3 Respect des bonnes pratiques et amélioration de la sécurité du service OpenSSH sur le serveur

- Mise en œuvre des bonnes pratiques édictées par l'ANSSI



Laboratoire 1

- Mise en place du laboratoire 1

Pour commencer la mise en place du laboratoire 1 il faut déjà ajouter le lieu d'installation de docker :

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/ap
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] htt
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Puis on installe docker :

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docke
```

Ensuite on télécharge le document gestion_lab1.sh :

```
root@TPthiryOUDARNicolas:~# wget https://forge.aeif.fr/btssio-labos-kali/lab1/-/raw/m
ain/gestion_lab1.sh -output-document gestion_lab1.sh
```

Pour la suite on lance le script :

```
root@TPthiryOUDARNicolas:~# bash gestion_lab1.sh -c
```

Pour finir on lance le laboratoire :

```
root@TPthiryOUDARNicolas:~#  bash gestion_lab1.sh -l

Démarrage des conteneurs

routeur-lab1
Conteneur routeur-lab1... Démarré.
kali-lab1
Conteneur kali-lab1... Démarré.
serveur-lab1
Conteneur serveur-lab1... Démarré.
client-lab1
Conteneur client-lab1... Démarré.
root@TPthiryOUDARNicolas:~#
```

Partie 1 Attaque MITM d'un service SSH

- Nmap

Q1 Pourquoi l'accès aux machines virtuelles par la console ou l'interface graphique n'est pas possible avec le super-administrateur root ?
L'accès en tant que super-administrateur root est évité par mesure de sécurité, car cela donne un contrôle total sur le système et peut causer des problèmes s'il est utilisé incorrectement. Les systèmes Linux préfèrent utiliser des comptes utilisateur standard pour éviter les erreurs.

Q2 Expliquer à quoi sert la commande sudo et quels avantages elle a sur l'utilisation de la commande « su - »

La commande "sudo" permet à un utilisateur d'exécuter des commandes d'administration avec des privilèges de super-administrateur temporaires, sans avoir besoin de se connecter en tant que super-administrateur.

Q3 Quelles commandes permettent de savoir si le service OpenSSH (serveur) est déjà installé et démarré ?

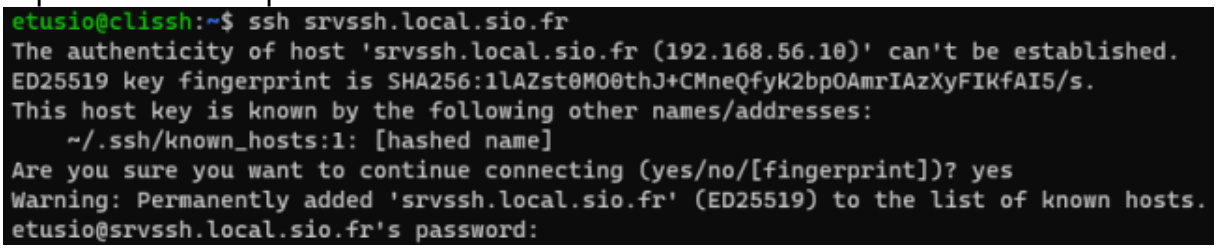
Avec la commande systemctl status sshd, si il est marqué comme active c'est qu'il est démarrer. Si la commander met une erreur c'est que le service n'est pas installé.

```
etusio@srvssh:~$ sudo systemctl status sshd
[sudo] Mot de passe de etusio :
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: enabled)
     Active: active (running) since Wed 2023-11-08 14:54:21 CET; 2h 52min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 51 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 57 (sshd)
      Tasks: 8 (limit: 697)
     Memory: 9.1M
```

Q4 Indiquer le répertoire où sont stockées les clés publique et privée créées ainsi que le positionnement des permissions appliquées sur les fichiers correspondants. Puis indiquer quel est le fichier de configuration du service SSH.

Les clés publiques et privées SSH sont généralement stockées dans le répertoire de l'utilisateur sous ~/.ssh/.
~ : Représente le dossier home de l'utilisateur courant Les fichiers ont la permission 600 qui signifie qu'ils sont en lecture seule par le propriétaire seulement. Le fichier de configuration du service ssh est /etc/ssh/sshd_config

Q5 Que signifie cette alerte qui est affichée à l'écran ? Devez-vous continuer l'opération ? Pourquoi ?



Cette alerte indique que la clé d'authentification du serveur SSH n'est pas reconnue sur ma machine. Il me donne l'empreinte digitale (fingerprint) de la clé ED25519 du serveur et me demande si je suis sûr de continuer.
On doit accepter si l'on veut pouvoir se connecter au serveur

Q6 Lors d'une prochaine connexion depuis le même client sur ce serveur, ce message apparaîtra-til à nouveau ? Pourquoi ?

Non. En acceptant la première fois, la clé est enregistrée dans le fichier known_hosts de l'utilisateur ce qui permet a ma machine de reconnaitre le serveur.

Q7 Sur la machine virtuelle cliente, expliquer à quoi sert le fichier /home/etusio/.ssh/known_hosts.

Le fichier ~/.ssh/known_hosts sert à stocker les empreintes digitales (fingerprints) des clés publiques des serveurs auxquels je me suis connecté précédemment. Grâce a ce fichier, le client ssh vérifier l'empreinte digitale de la clé du serveur distant par rapport à celle enregistrée dans ce fichier. Cela permet d'éviter les attaques comme le MITM (Man-In-The-Middle)

```
┌──(etusio㉿kali)-[~]
└─$ nmap -sP 192.168.56.0/24
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-08 15:12 CET
Nmap scan report for 192.168.56.1
Host is up (0.0065s latency).
Nmap scan report for serveur-lab1.bridge_interne_lab (192.168.56.10)
Host is up (0.0011s latency).
Nmap scan report for client-lab1.bridge_interne_lab (192.168.56.11)
Host is up (0.0065s latency).
Nmap scan report for kali (192.168.56.12)
Host is up (0.0051s latency).
Nmap scan report for routeur-lab1.bridge_interne_lab (192.168.56.254)
Host is up (0.00067s latency).
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.42 seconds
```

Je lance un premier scan du réseau afin d'y découvrir les appareils. Et ensuite je scan chacune des machines trouver dans le réseau

nmap -sV 192.168.56.10
nmap -sV 192.168.56.11
nmap -sV 192.168.56.254

```
┌──(etusio㉿kali)-[~]
└─$ nmap -sV 192.168.56.10
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-08 15:13 CET
Nmap scan report for serveur-lab1.bridge_interne_lab (192.168.56.10)
Host is up (0.00090s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.2p1 Debian 2 (protocol 2.0)
53/tcp open  domain  ISC BIND 9.18.16-1~deb12u1 (Debian Linux)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.76 seconds
```

Q8 Indiquer quelles sont les informations que peut obtenir un attaquant grâce à ces commandes ?
La première commande permet de faire un scan rapide du réseau grâce au paramètre -sP qui indique a nmap de ne pas scanner les ports. Ainsi il va scanner simplement le réseau en envoyant des paquets ICMP Echo Request à l'ensemble des adresses IP de la plage 192.168.56.0/24. (Il suffirait de bloquer les demandes ICMP avec un pare feu pour contrer ce type de détection)

La deuxième commande effectue une détection de services (-sV) sur les adresses IP trouver avec la commande précédente, en tentant d'identifier les services et leurs versions associées.

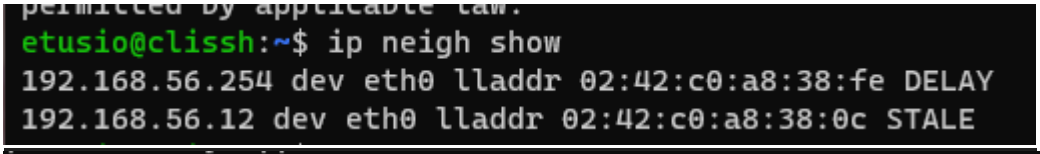
On y retrouve nos 3 debian (clissh, srvssh et router), sur chacune des machines on y retrouve un serveur SSH. Sur le serveur il y a un service DNS et sur le client il y a un service xRDP en plus

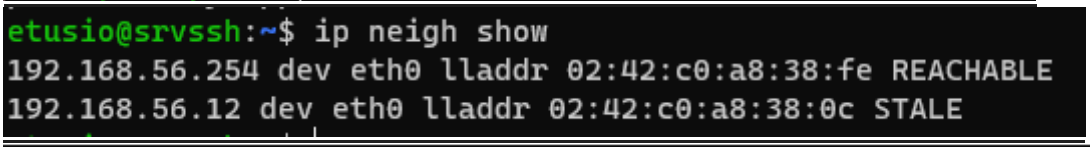
- L'homme du milieu (man in the middle)

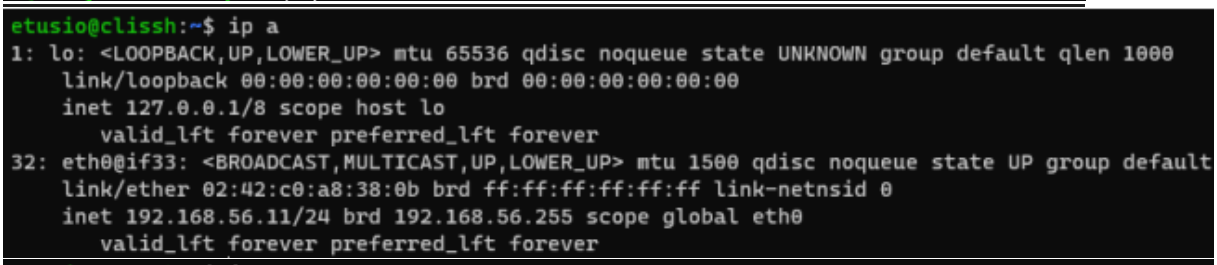
Q9 Expliquer les principes généraux d'une attaque de l'homme du milieu (Man in the Middle)

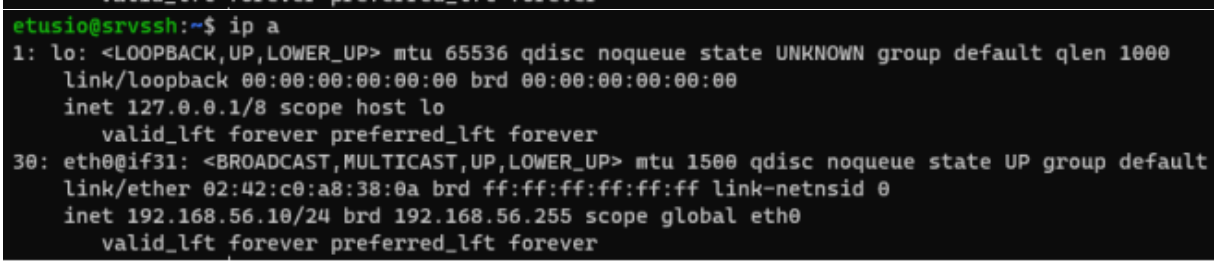
Une attaque de l'homme du milieu (Man-in-the-Middle ou MITM) est une forme d'attaque informatique où un attaquant s'insère secrètement entre deux parties qui communiquent, interceptant et éventuellement modifiant les données échangées entre elles.

Q10 Noter les associations adresse IP / adresse MAC présentes sur les deux machines. Sont-elles cohérentes ?









Sur le client (clissh) on peut y retrouver les adresses IP et MAC du serveur (srvssh) et du kali. En vérifiant l'adresse MAC des machines avec la commande ip a qui nous communiques des informations sur les interfaces réseau, on peut y voir que l'association IP / MAC est correcte.

Q11 Pourquoi l’activation du routage sur la machine de l’attaquant est indispensable au bon fonctionnement de l’attaque MITM ?

Si le routage n'est pas activé sur la machine de l'attaquant, le trafic ne sera pas redirigé vers les parties légitimes et la victime ne pourra pas communiquer avec le serveur. Donc il comprendra qu'il y a un problème.

Q12.Pourquoi cette redirection de ports est indispensable au succès de l’attaque de l’homme du milieu ?

La redirection de ports est indispensable au succès de l'attaque de l'homme du milieu car elle permet à l'attaquant d'intercepter tout le trafic destiné à la victime, même si celui-ci est chiffré.

En redirigeant le trafic vers son propre système, l'attaquant peut se placer en intermédiaire entre la victime et le serveur distant. Il peut alors lire, modifier ou supprimer les données transmises, sans que la victime ne s'en aperçoive.

Q14.Comparer les caches ARP du client et du serveur avec les associations notées précédemment lors de la question 10. Qu’en concluez-vous ?

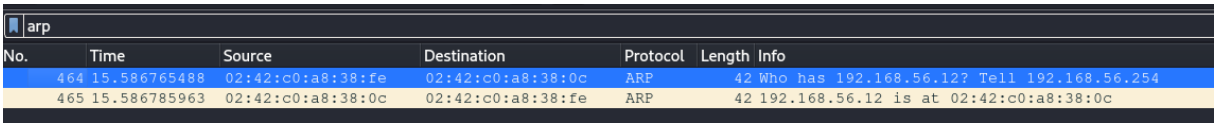
```
┌──(root㉿kali)-[/home/etusio/ssh-mitm]
└─# ettercap -i eth0 -T -M arp /192.168.56.10// /192.168.56.11//

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
  eth0 -> 02:42:C0:A8:38:0C
          192.168.56.12/255.255.255.0


SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...

  34 plugins
  42 protocol dissectors
  57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Scanning for merged targets (2 hosts)...

* |==================================================>| 100.00 %

2 hosts added to the hosts list...

ARP poisoning victims:

 GROUP 1 : 192.168.56.10 02:42:C0:A8:38:0A

 GROUP 2 : 192.168.56.11 02:42:C0:A8:38:0B
Starting Unified sniffing...


Text only Interface activated...
Hit 'h' for inline help
```

```
etusio@clissh:~$  ip neigh show
192.168.56.254 dev eth0 lladdr 02:42:c0:a8:38:fe DELAY
192.168.56.12 dev eth0 lladdr 02:42:c0:a8:38:0c STALE
192.168.56.10 dev eth0 lladdr 02:42:c0:a8:38:0c REACHABLE
etusio@clissh:~$
```

```
etusio@srvssh:~$  ip neigh show
192.168.56.254 dev eth0 lladdr 02:42:c0:a8:38:fe DELAY
192.168.56.12 dev eth0 lladdr 02:42:c0:a8:38:0c STALE
192.168.56.11 dev eth0 lladdr 02:42:c0:a8:38:0c REACHABLE
etusio@srvssh:~$
```

On peut remarquer qu'il y a une adresse IP en plus (le .10 sur le client et le .11 sur le serveur).

Q15.À partir de ces différentes observations, expliquer en détails comment fonctionne une attaque ARP Spoofing



Une attaque ARP Spoofing fonctionne en empoisonnant le cache ARP des machines sur le réseau. L'attaquant envoie des faux messages ARP aux machines, leur indiquant que son adresse MAC est associée à l'adresse IP de la machine cible. Les machines mettent alors à jour leur cache ARP et commencent à envoyer tout le trafic destiné à la machine cible à l'attaquant.

L'attaquant peut ensuite intercepter, modifier ou supprimer le trafic, sans que les machines victimes ne s'en aperçoivent.

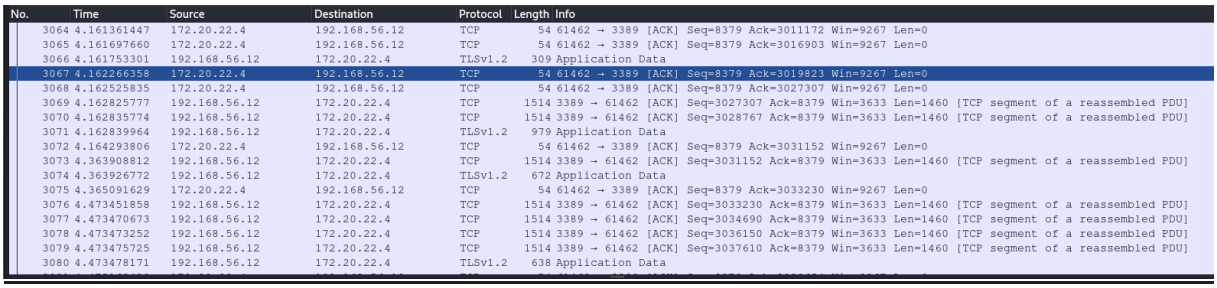
Exemple :

Un attaquant veut intercepter le trafic entre une machine client et un serveur SSH. Il envoie d'abord des faux messages ARP aux deux machines, leur indiquant que son adresse MAC est associée à l'adresse IP de l'autre machine.

Le client et le serveur mettent alors à jour leur cache ARP et commencent à envoyer tout le trafic destiné à l'autre machine à l'attaquant.

L'attaquant peut ensuite intercepter le trafic SSH et voler les identifiants de connexion de l'utilisateur.

Q16.Envoyer une requête ping (icmp-écho) depuis le client vers le serveur (192.168.56.10). Puis vérifier à l’aide d’une capture de trame sur la machine Kali Linux que ces dernières passent effectivement bien par l’attaquant. Quels éléments démontrent que l’attaque se déroule correctement ?



No.	Time	Source	Destination	Protocol	Length	Info
3064	4.161361447	172.20.22.4	192.168.56.12	TCP	54	61462 → 3389 [ACK] Seq=8379 Ack=3011172 Win=9267 Len=0
3065	4.161697660	172.20.22.4	192.168.56.12	TCP	54	61462 → 3389 [ACK] Seq=8379 Ack=3016903 Win=9267 Len=0
3066	4.161753301	192.168.56.12	172.20.22.4	TLSv1.2	309	Application Data
3067	4.162266358	172.20.22.4	192.168.56.12	TCP	54	61462 → 3389 [ACK] Seq=8379 Ack=3019823 Win=9267 Len=0
3068	4.162525835	172.20.22.4	192.168.56.12	TCP	54	61462 → 3389 [ACK] Seq=8379 Ack=3027307 Win=9267 Len=0
3069	4.162825777	192.168.56.12	172.20.22.4	TCP	1514	3389 → 61462 [ACK] Seq=3027307 Ack=8379 Win=3633 Len=1460 [TCP segment of a reassembled PDU]
3070	4.162835774	192.168.56.12	172.20.22.4	TCP	1514	3389 → 61462 [ACK] Seq=3028767 Ack=8379 Win=3633 Len=1460 [TCP segment of a reassembled PDU]
3071	4.162839964	192.168.56.12	172.20.22.4	TLSv1.2	979	Application Data
3072	4.164293806	172.20.22.4	192.168.56.12	TCP	54	61462 → 3389 [ACK] Seq=8379 Ack=3031152 Win=9267 Len=0
3073	4.363908812	192.168.56.12	172.20.22.4	TCP	1514	3389 → 61462 [ACK] Seq=3031152 Ack=8379 Win=3633 Len=1460 [TCP segment of a reassembled PDU]
3074	4.363926772	192.168.56.12	172.20.22.4	TLSv1.2	672	Application Data
3075	4.365091629	172.20.22.4	192.168.56.12	TCP	54	61462 → 3389 [ACK] Seq=8379 Ack=3033230 Win=9267 Len=0
3076	4.473451858	192.168.56.12	172.20.22.4	TCP	1514	3389 → 61462 [ACK] Seq=3033230 Ack=8379 Win=3633 Len=1460 [TCP segment of a reassembled PDU]
3077	4.473470673	192.168.56.12	172.20.22.4	TCP	1514	3389 → 61462 [ACK] Seq=3034690 Ack=8379 Win=3633 Len=1460 [TCP segment of a reassembled PDU]
3078	4.473473252	192.168.56.12	172.20.22.4	TCP	1514	3389 → 61462 [ACK] Seq=3036150 Ack=8379 Win=3633 Len=1460 [TCP segment of a reassembled PDU]
3079	4.473475725	192.168.56.12	172.20.22.4	TCP	1514	3389 → 61462 [ACK] Seq=3037610 Ack=8379 Win=3633 Len=1460 [TCP segment of a reassembled PDU]
3080	4.473478171	192.168.56.12	172.20.22.4	TLSv1.2	638	Application Data

Pour vérifier que les requêtes ping passent bien par l'attaquant, il faut rechercher des trames ICMP echo request avec l'adresse IP source de l'attaquant. Si on trouve de telles trames, cela signifie que l'attaque se déroule correctement.

Q17.Que contient le fichier /home/ssh-mitm/shell_session_0.txt présent sur Kali Linux ?

Le fichier `/home/ssh-mitm/shell_session_0.txt` sur Kali Linux contient l'ensemble des informations ayant transité entre le client et le serveur SSH lors d'une attaque de l'homme du milieu. Cela inclut les données suivantes :

- Le nom d'hôte et l'adresse IP du client et du serveur SSH
- Le nom d'utilisateur et le mot de passe utilisés pour s'authentifier sur le serveur SSH
- Toutes les commandes exécutées sur le serveur SSH

Ce fichier peut être utilisé par l'attaquant pour voler des informations sensibles, telles que des mots de passe ou des données de carte de crédit, ou pour prendre le contrôle du serveur SSH.

- Connexion SSH

Ensuite on vide le cache ARP avec la commande suivante et on essaye de se connecter en ssh au serveur :

```
etusio@clissh:~$ sudo ip neigh flush all
```

```
etusio@clissh:~$ ssh etusio@192.168.56.10
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:BzxsSdTtcC6DIPcSNd7g10cVa4E8DX/6HFLS3JmdlKE.
Please contact your system administrator.
Add correct host key in /home/etusio/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/etusio/.ssh/known_hosts:3
  remove with:
  ssh-keygen -f "/home/etusio/.ssh/known_hosts" -R "192.168.56.10"
Host key for 192.168.56.10 has changed and you have requested strict checking.
Host key verification failed.
```

Q18.Expliquer pourquoi ce message d'erreur apparaît.

Le message d'erreur "WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!" apparaît car la clé publique du serveur SSH a changé. Cela peut se produire pour plusieurs raisons, telles que :

- Le serveur SSH a été réinstallé.
- La clé publique du serveur SSH a été compromise.
- Un attaquant est en train de réaliser une attaque de l'homme du milieu (man-in-the-middle).

Il est important de prendre cette erreur au sérieux, car elle peut indiquer qu'un attaquant est en train de tenter de compromettre votre connexion SSH.

Q19.Proposer une solution afin de pouvoir à nouveau se connecter au service SSH depuis le client.

Pour pouvoir à nouveau se connecter au service SSH depuis le client, il existe deux solutions :

1. Vérifier l'empreinte de la clé publique du serveur SSH

Il est possible que la clé publique du serveur SSH ait changé. Pour le vérifier, il faut comparer l'empreinte de la clé affichée dans le message d'erreur avec l'empreinte de la clé publique connue. Si les deux empreintes sont différentes, cela signifie que la clé publique du serveur SSH a changé.

Pour obtenir l'empreinte de la clé publique du serveur SSH, on peut utiliser la commande suivante :

```
ssh-keygen -l -f ~/.ssh/known_hosts
```

Cette commande affichera une liste de toutes les clés publiques des serveurs SSH connus. Il faut rechercher la clé publique du serveur SSH auquel on essaye de se connecter et comparer l'empreinte de la clé avec l'empreinte de la clé affichée dans le message d'erreur.

Si les deux empreintes sont différentes, il faut contacter l'administrateur du serveur SSH pour s'assurer qu'il n'y a pas de problème de sécurité.

2. Supprimer la clé publique du serveur SSH du fichier known_hosts

Si l'administrateur du serveur SSH confirme que la clé publique du serveur SSH a changé, on peut supprimer la clé publique du fichier known_hosts. Cela permettra de se connecter au serveur SSH sans recevoir le message d'erreur.

Pour supprimer la clé publique du serveur SSH du fichier known_hosts, on peut utiliser la commande suivante :

```
ssh-keygen -R hostname
```

où `hostname` est le nom d'hôte ou l'adresse IP du serveur SSH.
Exemple

```
ssh-keygen -R srvssh.local.sio.fr
```

Cette commande supprimera la clé publique du serveur SSH `srvssh.local.sio.fr` du fichier known_hosts.

Une fois que la clé publique du serveur SSH a été supprimée du fichier known_hosts, on pourra se connecter au serveur SSH sans recevoir le message d'erreur.

Partie 2 Mise en place de contre-mesures

- Mise en place d'une authentification par clés de chiffrement

Pour mettre en place une authentification par clés de chiffrement il faut aller dans le fichier config de sshd puis autorisé les clés publiques puis ensuite mettre les fichiers de clés autorisé. :

```
PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2
```

Ensuite on restart ssh :

```
etusio@srvssh:~$ sudo service ssh restart
etusio@srvssh:~$  sudo systemctl restart sshd
```

Puis on génère une clés ssh ecdsa :

```
etusio@clissh:~$ ssh-keygen -b 256 -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/etusio/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/etusio/.ssh/id_ecdsa
Your public key has been saved in /home/etusio/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:msaS/iq1ueq2ycb1Vu9Tz/vtU6DCjCw5GxJatpvI46o etusio@clissh
The key's randomart image is:
+---[ECDSA 256]---+
|                 |
|                 |
|                 |
|    +         .  |
|   + o oS+   . . |
|  . =o=o+ + o   .|
| o +oB=* . o o  .|
| .Oo=o+   o   o..|
|E*B=+=.  ...  .+=|
+----[SHA256]-----+
```

Q1. Pourquoi l'algorithme ECDSA a été préféré à l'algorithme RSA lors de la génération de la paire de clés ?

L'algorithme ECDSA a été préféré à l'algorithme RSA pour la génération de la paire de clés car il offre plusieurs avantages, notamment :

- Une meilleure sécurité pour une même taille de clé. En effet, les clés ECDSA sont plus petites que les clés RSA pour un même niveau de sécurité.
- Une meilleure performance. Les opérations de signature et de vérification de signature sont plus rapides avec ECDSA qu'avec RSA.
- Une meilleure résistance aux attaques quantiques. Les algorithmes de cryptographie asymétrique basés sur les courbes elliptiques, tels qu'ECDSA, sont considérés comme plus résistants aux attaques quantiques que les algorithmes basés sur la factorisation de grands nombres, tels que RSA.

Q2. À votre avis, pourquoi la mise en place de cette phrase de chiffrement pour accéder à la clé privée est extrêmement importante ?

La mise en place d'une phrase de chiffrement pour accéder à la clé privée est extrêmement importante car elle permet de protéger la clé privée contre les accès non autorisés. En effet, si la clé privée est stockée en clair, elle peut être facilement volée par un attaquant. En revanche, si la clé privée est chiffrée avec une phrase de chiffrement, seul celui qui connaît la phrase de chiffrement pourra accéder à la clé privée.

Q3. Que faire de la paire de clés ? Si l'on récapitule, ssh-keygen a généré deux clés. Une clé privée qui est $HOME/.ssh/id_ecdsa à laquelle vous seul devez avoir accès et une clé publique qui est $HOME/.ssh/id_ecdsa.pub, qui peut être connue par tout le monde.

La clé privée doit être stockée en lieu sûr et ne doit être accessible à personne d'autre que vous. La clé publique peut être partagée avec les serveurs SSH auxquels vous souhaitez vous connecter.

Voici quelques conseils pour sécuriser votre clé privée :

- Stockez votre clé privée dans un endroit sûr, tel qu'un gestionnaire de mots de passe ou un disque dur chiffré.
- N'envoyez jamais votre clé privée par e-mail ou par SMS.
- Ne partagez votre clé privée avec personne.
- Utilisez une phrase de chiffrement forte pour protéger votre clé privée.

Voici quelques conseils pour utiliser votre clé publique :

- Partagez votre clé publique avec les serveurs SSH auxquels vous souhaitez vous connecter.
- Vous pouvez ajouter votre clé publique au fichier `known_hosts` de votre client SSH. Cela permettra à votre client SSH de vérifier l'empreinte de la clé publique du serveur SSH chaque fois que vous vous connectez.
- Vous pouvez également utiliser votre clé publique pour créer des certificats SSH.

Lister le contenu du répertoire $HOME/.ssh/ puis afficher le contenu du fichier id_ecdsa.pub.
Pour lister le contenu du répertoire `$HOME/.ssh/`, vous pouvez utiliser la commande suivante :

```
cd $HOME/.ssh/
ls
```

```
etusio@clissh:~/.ssh$ ls
id_ecdsa  id_ecdsa.pub  known_hosts  known_hosts.old
etusio@clissh:~/.ssh$
```

Pour afficher le contenu du fichier `id_ecdsa.pub`, vous pouvez utiliser la commande suivante :

```
cat id_ecdsa.pub
```

```
etusio@clissh:~/.ssh$ ssh-copy-id -i ~/.ssh/id_ecdsa.pub etusio@192.168.56.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/etusio/.ssh/id_e
cdsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out a
ny that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted no
w it is to install the new keys
etusio@192.168.56.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'etusio@192.168.56.10'"
and check to make sure that only the key(s) you wanted were added.
```

```
PasswordAuthentication no
```

```
etusio@clissh:~/.ssh$ ssh etusio@192.168.56.10
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for '/home/etusio/.ssh/id_ecdsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "/home/etusio/.ssh/id_ecdsa": bad permissions
etusio@192.168.56.10's password:
Linux srvssh 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_6
4

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: Wed Nov  8 15:39:09 2023 from 192.168.56.11
```

Q4. Sur la machine cliente clissh.local.sio.fr, modifier les droits du fichier ~/.ssh/id_ecdsa (droits initiaux : 600 etusio:etusio) qui contient votre clé privée en 644. Se connecter sur le serveur distant srvssh.local.sio.fr qui contient la clé publique. Que se passe-t-il ? Pourquoi ?

Sur la machine cliente clissh.local.sio.fr, modifier les droits du fichier ~/.ssh/id_ecdsa (droits initiaux : 600) qui contient votre clé privée en 644. Se connecter sur le serveur distant srvssh.local.sio.fr qui contient la clé publique. Que se passe-t-il ? Pourquoi ?

Lorsque vous modifiez les droits du fichier ~/.ssh/id_ecdsa de 600 à 644, vous permettez à tout utilisateur du système de lire la clé privée. Cela signifie que n'importe qui peut potentiellement utiliser votre clé privée pour se connecter à des serveurs SSH distants.

Si vous vous connectez sur le serveur distant srvssh.local.sio.fr qui contient la clé publique associée à votre clé privée, vous pourrez vous connecter sans avoir à saisir de mot de passe. En effet, le serveur SSH distant vérifiera la signature numérique de votre clé publique et, si elle est valide, il vous autorisera à vous connecter.

Pourquoi ?

SSH utilise la cryptographie asymétrique pour authentifier les utilisateurs. La cryptographie asymétrique utilise deux clés : une clé publique et une clé privée. La clé publique est utilisée pour chiffrer les données envoyées au serveur SSH distant,

tandis que la clé privée est utilisée pour déchiffrer les données reçues du serveur SSH distant.

Lorsque vous vous connectez à un serveur SSH distant, le client SSH envoie sa clé publique au serveur. Le serveur SSH distant vérifie ensuite la signature numérique de la clé publique. Si la signature numérique est valide, le serveur SSH distant autorise le client SSH à se connecter.

La signature numérique est une opération cryptographique qui permet de vérifier l'authenticité d'un message. Pour signer un message, le client SSH utilise sa clé privée. Seule la clé publique associée à la clé privée peut déchiffrer la signature numérique.

En modifiant les droits du fichier ~/.ssh/id_ecdsa de 600 à 644, vous permettez à tout utilisateur du système de lire la clé privée. Cela signifie que n'importe qui peut potentiellement utiliser votre clé privée pour signer des messages et se connecter à des serveurs SSH distants sans votre autorisation.

Conclusion
Il est important de garder votre clé privée secrète. Ne modifiez pas les droits du fichier ~/.ssh/id_ecdsa de 600 à 644. Si vous devez partager votre clé publique avec quelqu'un, vous pouvez utiliser la commande `ssh-copy-id`. Cette commande copiera votre clé publique sur le serveur SSH distant et la placera dans le fichier ~/.ssh/authorized_keys.

Q5. Rétablir les droits de ~/.ssh/id_ecdsa sur le poste client. Maintenant sur le serveur distant, afficher les droits d'accès appliqués au fichier ~/.ssh/authorized_keys.

```
etusio@clissh:~/.ssh$ chmod 600 id_ecdsa
etusio@clissh:~/.ssh$ ssh etusio@192.168.56.10
Linux srvssh 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_6
4

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: Wed Nov  8 16:12:01 2023 from 192.168.56.11
etusio@srvssh:~$ exit
```

Q6. Puis, modifier les droits de ~/.ssh/authorized_keys en 666. Se déconnecter puis se reconnecter. Vérifier si un changement est apparu ou non et tenter d'expliquer pourquoi (ne pas oublier de rétablir les droits d'origine ensuite).

```
etusio@srvssh:~/.ssh$ chmod 666 authorized_keys
```

Lorsque vous vous reconnectez, vous ne pourrez pas vous connecter sans saisir de mot de passe, même si vous avez ajouté votre clé publique au fichier `authorized_keys`.

En effet, le serveur SSH refuse de se connecter si les droits d'accès du fichier `authorized_keys` sont trop ouverts. Cela est dû à des raisons de sécurité, car un attaquant pourrait exploiter cette vulnérabilité pour accéder au serveur SSH sans autorisation.

Le serveur SSH vérifie les droits d'accès du fichier `authorized_keys` avant d'autoriser la connexion sans mot de passe. Si les droits d'accès sont trop ouverts, le serveur SSH refuse de se connecter.

Cela est dû à des raisons de sécurité. En effet, si un attaquant a accès au fichier `authorized_keys`, il peut ajouter sa propre clé publique au fichier et se connecter au serveur SSH sans autorisation.

Pour éviter cela, il est important de définir les droits d'accès du fichier `authorized_keys` à `644`. Cela signifie que seul le propriétaire du fichier peut lire et écrire le fichier.

Conclusion

Il est important de définir les droits d'accès du fichier `authorized_keys` à `644` pour des raisons de sécurité. Si vous définissez les droits d'accès à `666`, le serveur SSH refusera de se connecter.

Q7. En analysant la connexion par clés, proposer une hypothèse de fonctionnement de cette nouvelle forme d'authentification. Expliquer ce qui différencie une connexion par mot de passe d'une connexion par clé de chiffrement.

La connexion par clés est une forme d'authentification qui utilise une paire de clés de chiffrement pour authentifier un client à un serveur. La clé publique est stockée sur le serveur, tandis que la clé privée est stockée sur le client.

Lorsque le client se connecte au serveur, il envoie sa clé publique au serveur. Le serveur vérifie ensuite la signature numérique de la clé publique. Si la signature numérique est valide, le serveur autorise le client à se connecter.

La signature numérique est une opération cryptographique qui permet de vérifier l'authenticité d'un message. Pour signer un message, le client utilise sa clé privée. Seule la clé publique associée à la clé privée peut déchiffrer la signature numérique.

Différence entre une connexion par mot de passe et une connexion par clé de chiffrement

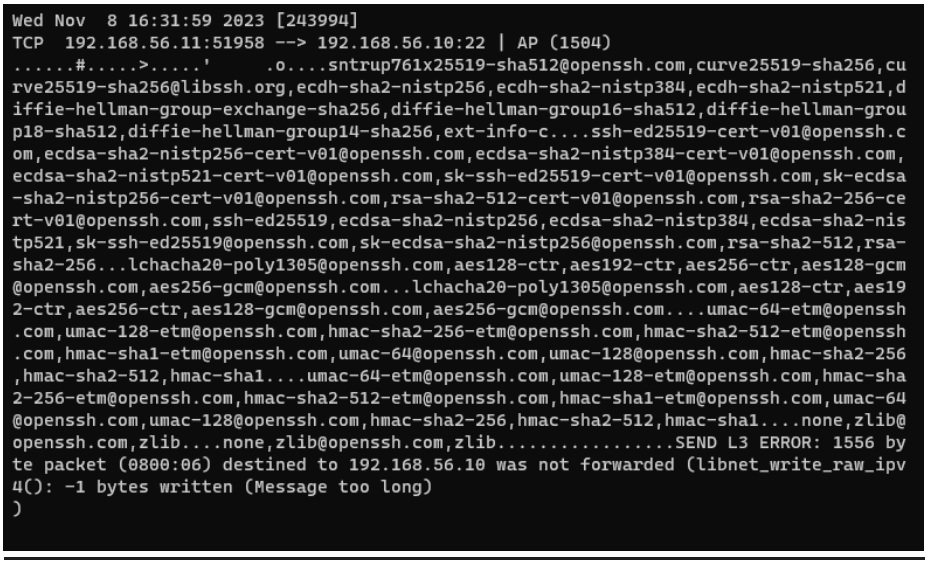
La principale différence entre une connexion par mot de passe et une connexion par clé de chiffrement est que la connexion par clé de chiffrement ne nécessite pas que l'utilisateur saisisse son mot de passe.

En effet, le client peut se connecter au serveur en envoyant simplement sa clé publique au serveur. Le serveur vérifie ensuite la signature numérique de la clé publique pour s'assurer que le client est autorisé à se connecter.

La connexion par clé de chiffrement est plus sécurisée que la connexion par mot de passe car elle est plus difficile à pirater. Un attaquant qui aurait accès à la clé publique ne pourrait pas se connecter au serveur sans également avoir accès à la clé privée.

Conclusion

La connexion par clés est une forme d'authentification plus sécurisée que la connexion par mot de passe car elle est plus difficile à pirater. Elle est également plus pratique pour l'utilisateur car elle ne nécessite pas de saisir de mot de passe.



- Utilisation de ssh-agent

Ensuite on utilise le ssh-agent :

```
etusio@clissh:~$  exec ssh-agent $SHELL
etusio@clissh:~$
```

```
etusio@clissh:~$ ssh-add
Identity added: /home/etusio/.ssh/id_ecdsa (etusio@clissh)
etusio@clissh:~$
```

Après avoir mis en place l'authentification par clés de chiffrement et activé l'agent ssh, une attaque MITM entre le client et le serveur SSH ne fonctionnera pas.

En effet, l'agent ssh stocke en mémoire la clé privée de l'utilisateur et la signe numériquement à chaque connexion. Le serveur SSH vérifie ensuite la signature numérique de la clé publique pour s'assurer que le client est autorisé à se connecter.

Un attaquant MITM ne serait pas en mesure de falsifier la signature numérique de la clé publique, même s'il était capable d'intercepter la connexion entre le client et le serveur SSH.

Conclusion

L'authentification par clés de chiffrement avec l'agent ssh est un moyen efficace de protéger les connexions SSH contre les attaques MITM.

- Faciliter la vérification de l'identité du serveur SSH avant la première connexion

```
etusio@srvssh:~$ ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256 SHA256:lPIxEKxsYHPP3i7iMiZZXLYUoW9viLwSfF39MNoWIM4 root@srvssh (ECDSA)
```

```
ssh etusio@srvssh.local.sio.fr
The authenticity of host ' srvssh.local.sio.fr (192.168.56.10)' can't be established.
ECDSA key fingerprint is SHA256:lPIxEKxsYHPP3i7iMiZZXLYUoW9viLwSfF39MNoWIM4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'srvssh.local.sio.fr, 192.168.56.10' (ECDSA) to the list of known hosts.
```

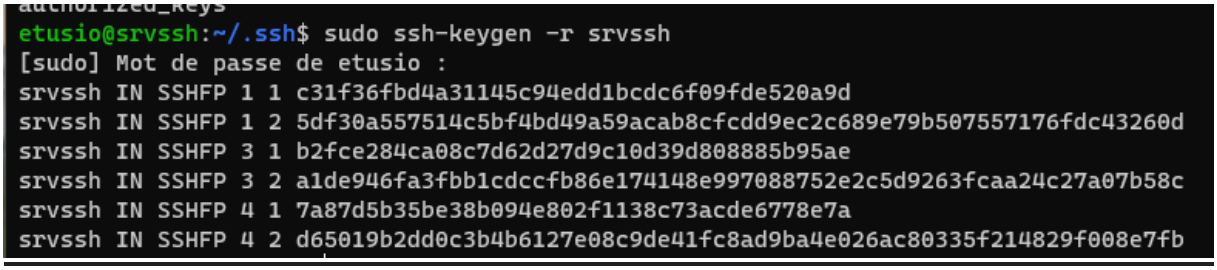
Q9. Expliquer l’intérêt de cette démarche.

La publication des empreintes de clés publiques du serveur SSH dans la zone DNS de l'entreprise présente plusieurs intérêts :

- Simplicité : les utilisateurs n'ont plus à ajouter manuellement la clé du serveur SSH à leur fichier `known_hosts`.
- Sécurité : les utilisateurs sont protégés contre les attaques de type "man-in-the-middle". En effet, si un attaquant tente de se faire passer pour le serveur SSH, l'empreinte de clé publique ne correspondra pas à celle publiée dans la zone DNS.
- Centralisation : la gestion des clés SSH est centralisée au niveau du serveur DNS, ce qui facilite la maintenance et la sécurité.

Conclusion

La publication des empreintes de clés publiques du serveur SSH dans la zone DNS de l'entreprise est une démarche simple, sécurisée et centralisée qui permet de protéger les utilisateurs contre les attaques de type "man-in-the-middle".



La commande sudo ssh-keygen -r srvssh supprime la paire de clés SSH du serveur srvssh de l'hôte local. Ceci est utile si vous n'avez plus besoin d'accéder au serveur ou si vous pensez que la paire de clés a été compromise.

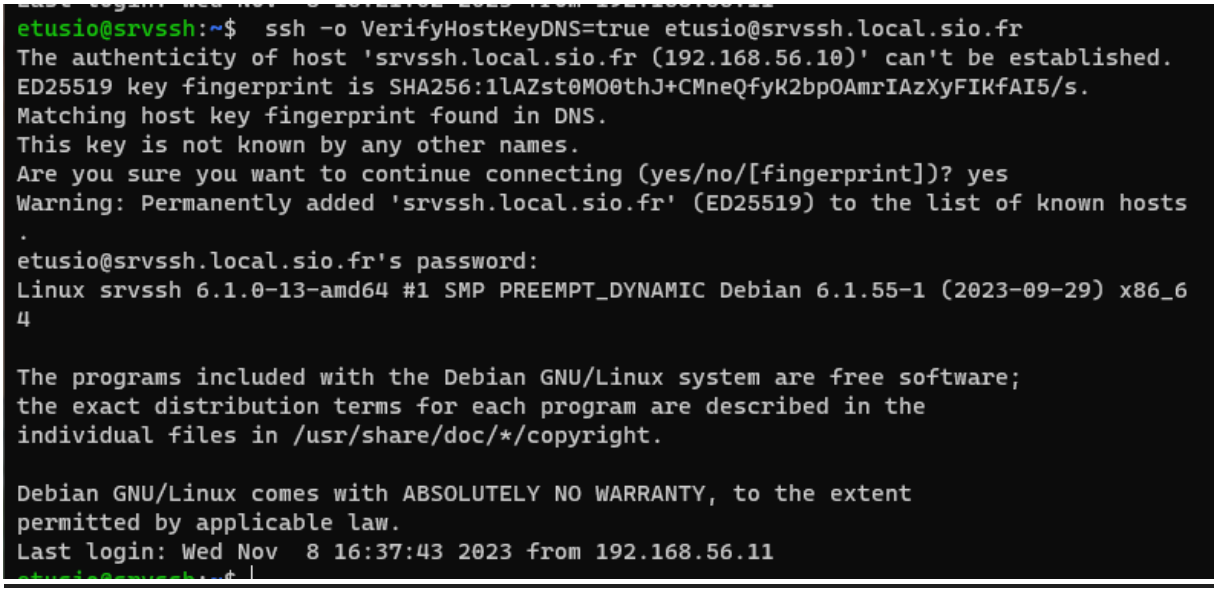
Pour utiliser la commande, vous devez disposer des privilèges sudo. En effet, la commande supprime les fichiers système.

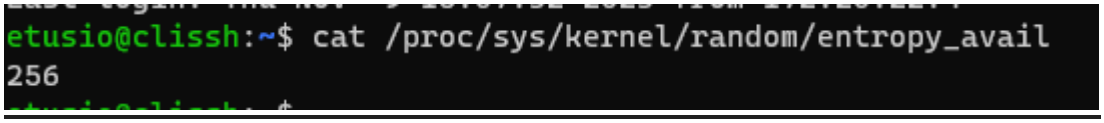
Une fois la commande exécutée, la paire de clés SSH pour srvssh sera supprimée des emplacements suivants :

/home/etusio/.ssh/known_hosts
/home/etusio/.ssh/config

Si vous essayez de vous connecter à srvssh après avoir exécuté la commande, vous serez invité à accepter l'empreinte digitale de la clé du serveur. En effet, l'hôte local n'aura plus d'enregistrement de la paire de clés du serveur.

Puis on fait une vérification :





Partie 3 Respect des bonnes pratiques et amélioration de la sécurité du service OpenSSH sur le serveur

- Mise en œuvre des bonnes pratiques édictées par l'ANSSI

Q1. Mettre en œuvre les préconisations suivantes tirées des recommandations pour un usage sécurisé de SSH publié par l'ANSSI :

1. Vérifier que les clés privées de chiffrement présentes dans le répertoire /etc/ssh/ appartiennent à l'utilisateur root en lecture-écriture seulement ;

2. S’assurer que c’est bien la version 2 du protocole SSH qui est utilisée ;

```
debug1: Local version string SSH-2.0-OpenSSH_9.2p1 Debian-2
debug1: Remote protocol version 2.0, remote software version OpenSSH_9.2p1 Debian-2
debug1: compat_banner: match: OpenSSH_9.2p1 Debian-2 pat OpenSSH* compat 0x04000000
```

3. Le serveur SSH doit dorénavant écouter sur le port 222/TCP ;

```
  GNU nano 7.2                                 sshd_config

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 222
#AddressFamily any
#ListenAddress 0.0.0.0
```

4. Vérifier que les droits sur les fichiers sont appliqués de manière stricte par SSH ;

```
-rw-r--r-- 1 root root 573928  8 févr.  2023 moduli
-rw-r--r-- 1 root root   1650  8 févr.  2023 ssh_config
```

5. L’accès SSH par l’utilisateur root doit être interdite ;

```
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
```

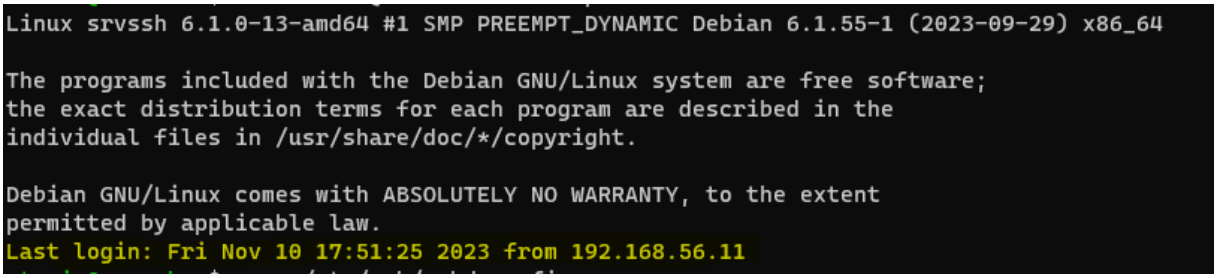
6. Mettre en œuvre une séparation des privilèges à l’aide d’un bac à sable (sandbox) ;

```
#MaxSessions 10
UsePrivilegeSeparation yes
```

7. L’accès à distance par des comptes ne disposant pas de mot de passe doit être interdit ;

8. Autoriser 3 tentatives de connexion successives en cas d’erreur dans le mot de passe ;

9. Le service doit afficher les informations de dernière connexion à l’utilisateur quand il se connecte ;



Q2. Expliquer dans une définition succincte ce qu’est l’état de l’art dans le domaine de la cybersécurité.

Définition succincte de l'état de l'art dans le domaine de la cybersécurité :
Ensemble des bonnes pratiques, des technologies et des documents de référence relatifs à la sécurité des systèmes d'information, qui permettent de protéger les systèmes et les données contre les cyberattaques.

Q3. Définir ce qu’est le principe de Kerckhoffs.

Le principe de Kerckhoffs est un principe de cryptographie selon lequel la sécurité d'un cryptosystème doit reposer uniquement sur le secret de la clé, et non sur le secret de l'algorithme. En d'autres termes, un cryptosystème doit être suffisamment robuste pour résister à une attaque même si l'adversaire connaît l'algorithme utilisé.

Ce principe est important car il permet de concevoir des cryptosystèmes plus sûrs et plus faciles à analyser. En effet, si la sécurité d'un cryptosystème repose sur le secret de l'algorithme, il est très difficile de vérifier que cet algorithme est

effectivement sûr. De plus, si le secret de l'algorithme est compromis, le cryptosystème est immédiatement compromis.

Le principe de Kerckhoffs a été formalisé par Auguste Kerckhoffs en 1883 dans son ouvrage "La cryptographie militaire". Il est aujourd'hui l'un des principes fondamentaux de la cryptographie moderne.

Q4. Selon ce principe, pourquoi est-il pertinent de choisir des algorithmes cryptographiques connus et respectant l'état de l'art ?

Selon le principe de Kerckhoffs, la sécurité d'un cryptosystème doit reposer uniquement sur le secret de la clé, et non sur le secret de l'algorithme. Par conséquent, il est pertinent de choisir des algorithmes cryptographiques connus et respectant l'état de l'art pour les raisons suivantes :

- Les algorithmes connus sont plus faciles à analyser. Plus un algorithme est connu et utilisé, plus il a été analysé par les cryptologues. Cela signifie que les éventuelles failles de l'algorithme sont plus susceptibles d'être découvertes et corrigées.
- Les algorithmes respectant l'état de l'art sont plus sûrs. Les algorithmes cryptographiques évoluent constamment, et les nouveaux algorithmes sont conçus pour résister aux attaques les plus récentes. En choisissant un algorithme respectant l'état de l'art, on augmente les chances que l'algorithme soit capable de résister aux attaques futures.
- Les algorithmes connus et respectant l'état de l'art sont plus interopérables. Les algorithmes cryptographiques sont utilisés dans une grande variété d'applications, et il est important que ces applications puissent communiquer entre elles de manière sécurisée. En choisissant des algorithmes connus et respectant l'état de l'art, on augmente la probabilité que les applications puissent communiquer entre elles en utilisant les mêmes algorithmes.

Pour empêcher l'usage d'algorithmes de chiffrement dépréciés, il est nécessaire d'éditer le fichier de configuration du serveur SSH et ajouter les lignes suivantes :

etusio@srvssh:~$ sudoedit /etc/ssh/sshd_config
KexAlgorithms curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-
nistp256,diffie-hellman-group-exchange-sha256
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-

gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com