Laboratoire 1

Mise en place du laboratoire

Sommaire :

Informations sur les machines et leurs configuration réseau :

Machine	Nom de domaine	Configuration réseau	Applications et services
Serveur sous Debian 12	srvssh.local.sio.fr	Adresse IPv4 : 192.168.56.10/24 Passerelle : 192.168.56.254 Serveur DNS : 192.168.56.10	Service OpenSSH port 22/TCP Service DNS Bind port 53/UDP
Client sous Debian 12	clissh.local.sio.fr	Adresse IPv4 : 192.168.56.11/24 Passerelle : 192.168.56.254 Serveur DNS : 192.168.56.10	Environnement de bureau XFCE Service XRDP port 3389/TCP Client OpenSSH
Attaquant sous Kali Linux	kali.local.sio.fr	Adresse IPv4 : 192.168.56.12/24 Passerelle :192.168.56.254 Serveur DNS :192.168.56.10	Environnement de bureau XFCE Service XRDP port 3389/TCP Ettercap Git ssh-mitm Netfilter/Iptables
Routeur sous Debian 12	routeur.local.sio.fr	Adresses IPv4 : eth0 – DHCP eth1 - 192.168.56.254/24 Serveur DNS : 192.168.56.10	Netfilter/Iptables

Intitulé de la machine	Nom d'utilisateur	Mot de passe	Ports SSH	Port RDP
Serveur SSH sous Debian 12	etusio	Fghijkl1234*	12222	
Client SSH sous Debian 12	etusio	Fghijkl1234*	22222	23389
Attaquant sous Kali Linux 2023.3	etusio	Fghijkl1234*	32222	33389
Routeur sous Debian 12	etusio	Fghijkl1234*	42222	

Mise en œuvre de la maquette :

Cette maquette sera réalisée sur une machine virtuelle Debian12.

Avant quelconque manipulations, je vérifie que ma machine est bien à jour à l'aide des commande apt-get update et apt-get upgrade.

```
root@debian:~# apt-get update
Atteint :1 http://deb.debian.org/debian bookworm InRelease
Atteint :2 http://security.debian.org/debian-security bookworm-security InRelea
e
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets... Fait
root@debian:~# apt-get upgrade
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Calcul de la mise à jour... Fait
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
root@debian:~#
```

J'installe ensuite les paquets nécessaires à l'utilisation du dépôt docker.

« apt install ca-certificates curl gnupg »

```
root@debian:~# apt install ca-certificates curl gnupg
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
ca-certificates est déjà la version la plus récente (20230311).
gnupg est déjà la version la plus récente (2.2.40-1.1).
gnupg passé en « installé manuellement ».
Les NOUVEAUX paquets suivants seront installés :
  curl
0 mis à jour, 1 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 315 ko dans les archives.
Après cette opération, 500 ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] o
Réception de :1 http://security.debian.org/debian-security bookworm-security/mai
n amd64 curl amd64 7.88.1-10+deb12u4 [315 kB]
315 ko réceptionnés en 0s (4 562 ko/s)
Sélection du paquet curl précédemment désélectionné.
(Lecture de la base de données... 154743 fichiers et répertoires déjà installés.
```

J'importe ensuite de la clé du dépôt docker.

« curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg chmod a+r /etc/apt/keyrings/docker.gpg »

```
root@debian:~# curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
chmod a+r /etc/apt/keyrings/docker.gpg
Le fichier « /etc/apt/keyrings/docker.gpg » existe. Faut-il réécrire par-dessus ? (o/N) o
```

J’intègre ensuite le dépôt dans le fichier source.list puis je mets à jour les dépôts avec la commande apt update.

```
root@debian:~# echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

La variable « $(. /etc/os-release; echo "$ID") » renvoie la distribution (ici « debian »).

La variable $(dpkg –print-architecture) renvoie l’architecture du processeur (ici amd64).

Après cela, je vais maintenant installer Docker.

« apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin »

```
root@debian:~# apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  docker-ce-rootless-extras git git-man iptables liberror-perl libip6tc2 libslirp0 patch pigz slirp4netns
Paquets suggérés :
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn firewalld ed diffutils-doc
Les NOUVEAUX paquets suivants seront installés :
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin git git-man
  iptables liberror-perl libip6tc2 libslirp0 patch pigz slirp4netns
0 mis à jour, 15 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 124 Mo dans les archives.
Après cette opération, 459 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] o
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 pigz amd64 2.6-1 [64,0 kB]
```

Ici, je modifie les droits de l’utilisateur en l’ajoutant au groupe docker afin de ne pas avoir besoin d’utiliser docker en administrateur.

```
root@debian:~# gpasswd -a root docker
Ajout de l'utilisateur root au groupe docker
```

Je récupère ensuite le script du lab1.

« git clone https://forge.aeif.fr/btssio-labos-kali/lab1.git »

```
root@debian:~# git clone https://forge.aeif.fr/btssio-labos-kali/lab1.git
Clonage dans 'lab1'...
remote: Enumerating objects: 141, done.
remote: Counting objects: 100% (134/134), done.
remote: Compressing objects: 100% (134/134), done.
remote: Total 141 (delta 76), reused 0 (delta 0), pack-reused 7
Réception d'objets: 100% (141/141), 771.91 Kio | 9.08 Mio/s, fait.
Résolution des deltas: 100% (77/77), fait.
```

Docker est installé, je vérifie son statut. On voit que Docker est bien actif.

« systemctl status docker »

```
root@debian:~# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Tue 2023-11-07 12:26:33 CET; 4min 4s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 5237 (dockerd)
      Tasks: 8
     Memory: 29.5M
        CPU: 164ms
     CGroup: /system.slice/docker.service
             └─5237 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Ici, je créer les images personnalisées des machines KALI, SEVEUR, CLIENT et ROUTEUR.

<u>Machine KALI :</u>

« bash gestion_lab1.sh -i KALI »

```
root@debian:~/lab1# bash gestion_lab1.sh -i KALI
Image est reseaucerta/kalirolling:lab1
Dockerfile à utiliser est dockerfile_KALI
[+] Building 917.8s (20/20) FINISHED                                                                                docker:d
 => [internal] load build definition from dockerfile_KALI
 => => transferring dockerfile: 4.58kB
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/kalilinux/kali-rolling:latest
 => [ 1/16] FROM docker.io/kalilinux/kali-rolling@sha256:38afc8fac34dc904a37f12e61566e2f268ddd81bfcb41dd2588818cdc68677a3
 => => resolve docker.io/kalilinux/kali-rolling@sha256:38afc8fac34dc904a37f12e61566e2f268ddd81bfcb41dd2588818cdc68677a3
 => => sha256:79d8b0ff606c3f2a4fe66ef10665705480ecd66368f84d34951a89429ccfe293 51.40MB / 51.40MB
 => => sha256:38afc8fac34dc904a37f12e61566e2f268ddd81bfcb41dd2588818cdc68677a3 1.19kB / 1.19kB
 => => sha256:f0817278153c7ae7a148dcf867b81d0270582d31d54a6d7390c241a5cd61bcfa 429B / 429B
 => => sha256:3e1e8d6278e825c2a2068e454435bcef70ba71fe3fbd397f7e4663d57c166384 2.85kB / 2.85kB
```

<u>Machine SERVEUR :</u>

Bash gestion_lab1.sh -i SERVEUR

```
root@debian:~/lab1# bash gestion_lab1.sh -i SERVEUR
Image est reseaucerta/serveurdebian12:lab1
Dockerfile à utiliser est dockerfile_SERVEUR
[+] Building 39.6s (12/12) FINISHED                                                                                 docker:de
 => [internal] load build definition from dockerfile_SERVEUR
 => => transferring dockerfile: 1.24kB
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/reseaucerta/basedebian12:1.0
 => [1/7] FROM docker.io/reseaucerta/basedebian12:1.0@sha256:1dae92eba26e28a462a9b92ee9775083cf4ecf61654b9270a42f876c9f7e
 => => resolve docker.io/reseaucerta/basedebian12:1.0@sha256:1dae92eba26e28a462a9b92ee9775083cf4ecf61654b9270a42f876c9f7e9
 => => sha256:1dae92eba26e28a462a9b92ee9775083cf4ecf61654b9270a42f876c9f7e91ab 1.78kB / 1.78kB
```

Machine CLIENT :

Bash gestion_lab1.sh -i CLIENT

```
root@debian:~/lab1# bash gestion_lab1.sh -i CLIENT
Image est reseaucerta/clientdebian12:lab1
Dockerfile à utiliser est dockerfile_CLIENT
[+] Building 89.9s (11/11) FINISHED                                                                                    docker:de
 => [internal] load build definition from dockerfile_CLIENT
 => => transferring dockerfile: 2.55kB
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/reseaucerta/basedebian12:1.0
 => CACHED [1/7] FROM docker.io/reseaucerta/basedebian12:1.0@sha256:1dae92eba26e28a462a9b92ee9775083cf4ecf61654b9270a42f87
 => [2/7] RUN apt update -q --fix-missing     && apt upgrade -y     && DEBIAN_FRONTEND=noninteractive apt -y install --no
 => [3/7] RUN rm /etc/localtime     && ln -s /usr/share/zoneinfo/Europe/Paris /etc/localtime     && dpkg-reconfigure --fro
 => [4/7] RUN systemctl enable xrdp.service
 => [5/7] RUN adduser xrdp ssl-cert     && echo xfce4-session > /home/etusio/.xsession     && chown etusio.etusio /home/et
 => [6/7] RUN rm /etc/xdg/autostart/xfce4-power-manager.desktop     && if [ -e /etc/xdg/xfce4/panel/default.xml ] ;     th
```

Machine ROUTEUR :

Bash gestion_lab1.sh -i ROUTEUR

```
root@debian:~/lab1# bash gestion_lab1.sh -i ROUTEUR
Image est reseaucerta/routeurdebian12:lab1
Dockerfile à utiliser est dockerfile_ROUTEUR
[+] Building 43.1s (6/6) FINISHED                                                                                      docker:de
 => [internal] load build definition from dockerfile_ROUTEUR
 => => transferring dockerfile: 787B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/reseaucerta/basedebian12:1.0
 => CACHED [1/2] FROM docker.io/reseaucerta/basedebian12:1.0@sha256:1dae92eba26e28a462a9b92ee9775083cf4ecf61654b9270a42f87
 => [2/2] RUN apt update -q --fix-missing     && apt upgrade -y     && DEBIAN_FRONTEND=noninteractive apt -y install --no
 => exporting to image
 => => exporting layers
 => => writing image sha256:e09e5eec46217668457f19b6505c8f38239f617affafaefb8af0137fb112fc0b
 => => naming to docker.io/reseaucerta/routeurdebian12:lab1
```

Je vérifie maintenant que je puisse bien communiqué avec les machines à l'aide d'un ping pour chacune d'elle à partir de la machine Debian.

Ping Debian vers la machine Serveur :

```
root@debian:~# ping 192.168.56.10
PING 192.168.56.10 (192.168.56.10) 56(84) bytes of data.
64 bytes from 192.168.56.10: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 192.168.56.10: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 192.168.56.10: icmp_seq=3 ttl=64 time=0.043 ms
^C
--- 192.168.56.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.043/0.055/0.080/0.017 ms
```

Ping Debian vers la machine Client :

```
root@debian:~# ping 192.168.56.11
PING 192.168.56.11 (192.168.56.11) 56(84) bytes of data.
64 bytes from 192.168.56.11: icmp_seq=1 ttl=64 time=0.075 ms
64 bytes from 192.168.56.11: icmp_seq=2 ttl=64 time=0.041 ms
64 bytes from 192.168.56.11: icmp_seq=3 ttl=64 time=0.046 ms
^C
--- 192.168.56.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.041/0.054/0.075/0.015 ms
```

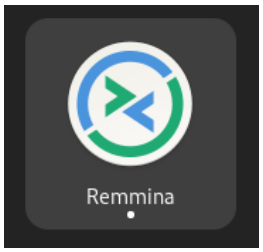
Ping Debian vers la machine Routeur :

```
root@debian:~# ping 192.168.56.254
PING 192.168.56.254 (192.168.56.254) 56(84) bytes of data.
64 bytes from 192.168.56.254: icmp_seq=1 ttl=64 time=0.077 ms
64 bytes from 192.168.56.254: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 192.168.56.254: icmp_seq=3 ttl=64 time=0.050 ms
^C
--- 192.168.56.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.050/0.059/0.077/0.012 ms
```

Ping Debian vers la machine KALI :

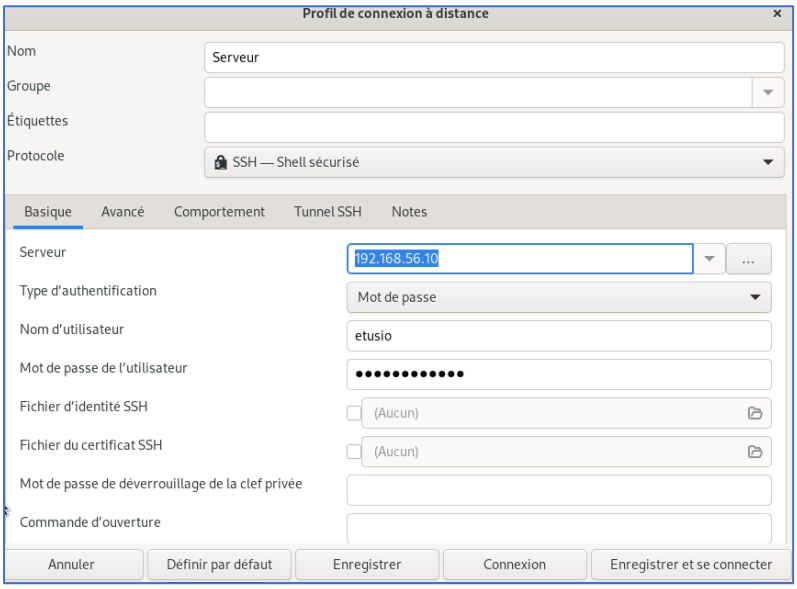
```
root@debian:~# ping 192.168.56.12
PING 192.168.56.12 (192.168.56.12) 56(84) bytes of data.
64 bytes from 192.168.56.12: icmp_seq=1 ttl=64 time=0.077 ms
64 bytes from 192.168.56.12: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 192.168.56.12: icmp_seq=3 ttl=64 time=0.049 ms
^C
--- 192.168.56.12 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2044ms
rtt min/avg/max/mdev = 0.049/0.058/0.077/0.013 ms
```

Pour accéder aux interfaces graphiques de mes machines, j'installe le logiciel Remina.

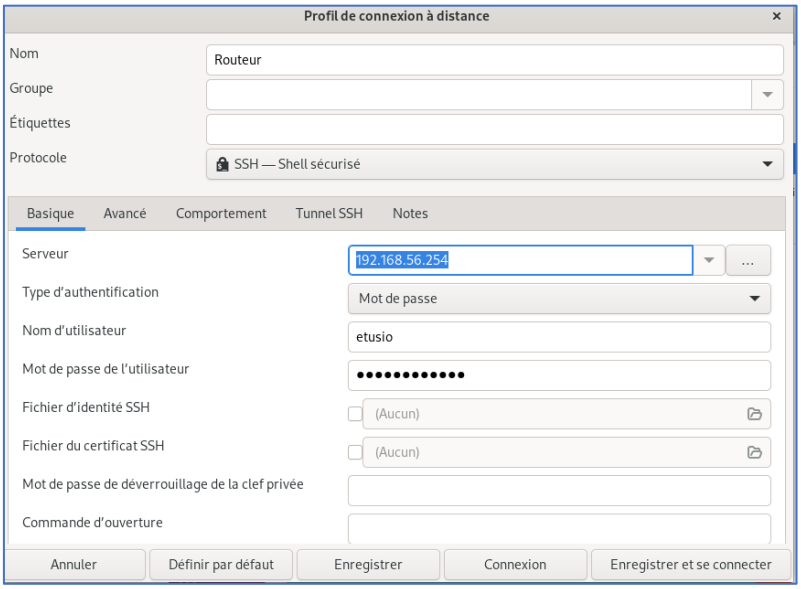


Sur Remmina, j'ajoute les machines à l'aide des informations disponible dans les tableaux de la page 2.

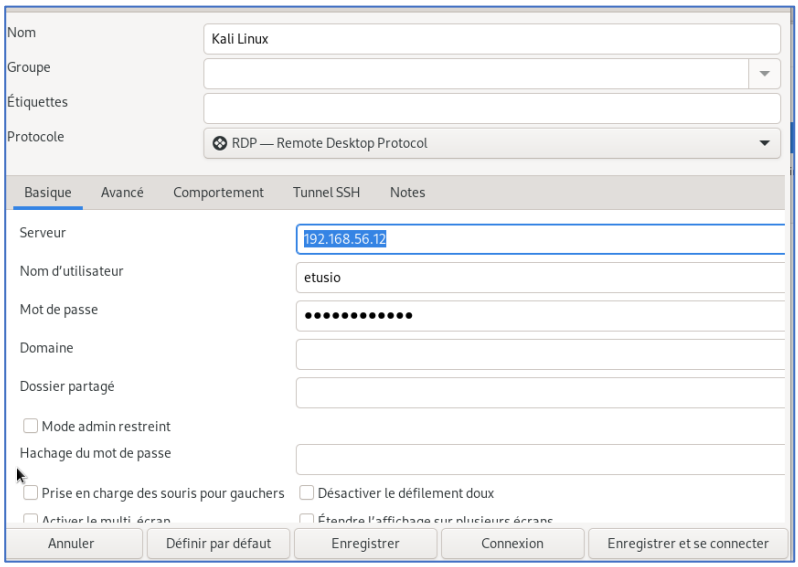
Serveur :



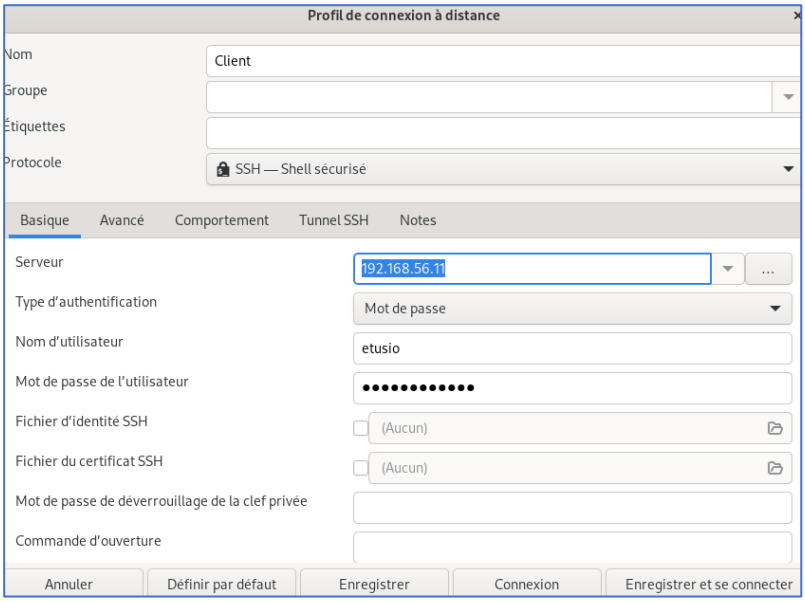
Routeur :



Kali :



Client :



Les machines sont maintenant disponibles sur Remmina.

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

Partie 1 – Attaque MITM d'un service SSH

Généralités

- 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 aux machines virtuelles par la console ou l'interface graphique n'est pas possible pour le super-administrateur root en raison de problèmes de sécurité et de gestion. Cela évite les risques de compromettre la sécurité, d'altérer l'isolation des machines virtuelles, de gérer efficacement les ressources et de respecter les politiques de gestion des utilisateurs.

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

La commande ***sudo*** permet aux utilisateurs d'exécuter des commandes avec des privilèges d'administration tout en restant authentifiés en tant qu'utilisateurs ordinaires. Les avantages de ***sudo*** par rapport à la commande ***su -*** incluent une gestion plus précise des privilèges, une journalisation des actions, une sécurité renforcée, une gestion des utilisateurs simplifiée, un contrôle d'accès basé sur des politiques et la préservation de la confidentialité du mot de passe root.

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

Pour voir si le service OpenSSH(Serveur) est installé/démarré, nous devons utilisé la commande ***service start ssh***.

Pour l'installer, nous utilisons la commande ***apt install openssh-server***

S'il est démarré, il sera actif sinon il sera offline.

Pour l'activer, nous devons utilisé la commande ***service ssh start***

- 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 SSH sont stockées dans le répertoire personnel de l'utilisateur sous ~/.ssh. La clé privée se trouve généralement dans un fichier tel que id_rsa, et la clé publique dans id_rsa.pub. Les permissions des fichiers sont généralement définies de manière sécurisée, avec la clé privée en lecture seule pour l'utilisateur et aucune autorisation pour les autres utilisateurs. Pour le service SSH, le fichier de configuration principal se trouve généralement dans /etc/ssh/sshd_config.

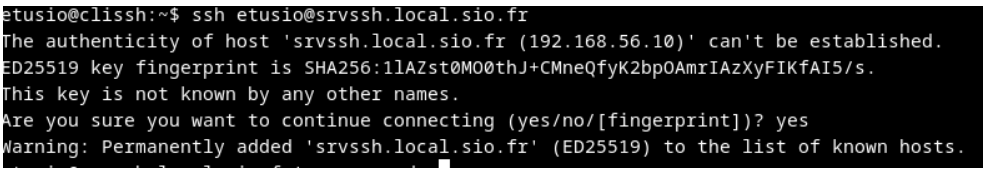
Première utilisation

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

L'alerte signifie que l'authenticité de l'hôte distant n'a pas encore été établie sur le système. Il faut comparer la clé de chiffrement soit comparé avec les clés présentes dans le fichier « known_hosts », cependant le fichier n'existe pas encore et sera créer lors de la première connexion en ssh.

Je réponds donc « yes » pour que continuer la connexion et donc créer le fichier et la clé de chiffrement.

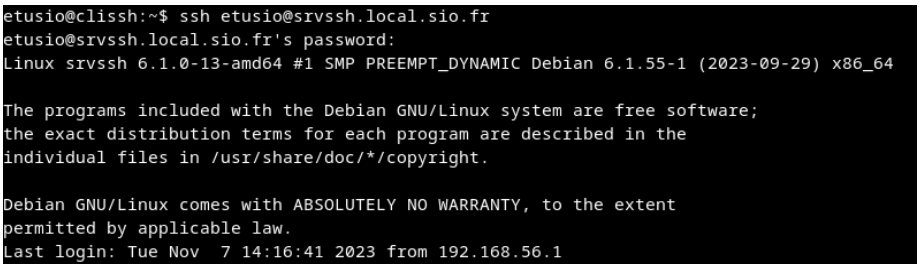
Le fichier « known_hosts » n'est pas un fichier existant sur la machine client. Ce fichier se créer lors de la première connexion en ssh réalisé au part avant.



- Q6. Lors d'une prochaine connexion depuis le même client sur ce serveur, ce message apparaîtra-t-il à nouveau ? Pourquoi ?

Lors de la prochaine connexion depuis le même client sur ce serveur ce message n'apparaîtra plus car la clé a été créée. Cela signifie que lors de la connexion en ssh, le client compare les clés contenues dans le fichier « known_hosts ». La clé est présente dans ce fichier donc le message n'apparaîtra pas.

Je supprime le contenu du fichier à l'aide de la commande echo > ~/.ssh/known_hosts.



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

Le fichier « /home/etusio/.ssh/known_hosts » stocke les clés d'authentifications des hôtes distants auxquels on s'est connecté précédemment via SSH. Il est utilisé pour vérifier l'authenticité des hôtes lors des connexions futures.

- Q8. Indiquer quelles sont les informations que peut obtenir un attaquant grâce à ces commandes ?

La commande nmap -sV 192.168.56.11 exécutée sur Kali Linux permet à un attaquant d'obtenir des informations sur l'hôte cible, notamment la liste des ports ouverts, les services en cours d'exécution, les versions des services et éventuellement des informations sur le système d'exploitation. Ces informations peuvent être utilisées à des fins de reconnaissance ou d'attaques, mais leur utilisation doit être légale et éthique.

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

Une attaque de l'homme du milieu (MITM) est une cyberattaque où un attaquant s'insère entre deux parties qui communiquent, intercepte leurs données, pour se faire passer pour l'une des parties, modifier les données en transit et capturer des informations sensibles. L'objectif est de compromettre la confidentialité et l'intégrité de la communication.

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

Oui les adresses sont cohérentes, on peut voir que les adresse IP et MAC correspondent bien entre eux.

Client :

```
root@clissh:/home/etusio# ip neigh show
192.168.56.1 dev eth0 lladdr 02:42:12:73:e2:cb REACHABLE
192.168.56.254 dev eth0 lladdr 02:42:c0:a8:38:fe STALE
192.168.56.10 dev eth0 lladdr 02:42:c0:a8:38:0c REACHABLE
192.168.56.12 dev eth0 lladdr 02:42:c0:a8:38:0c STALE
```

Serveur :

```
etusio@srvssh:~$ ip neigh show
192.168.56.11 dev eth0 lladdr 02:42:c0:a8:38:0c REACHABLE
192.168.56.1 dev eth0 lladdr 02:42:12:73:e2:cb REACHABLE
192.168.56.254 dev eth0 lladdr 02:42:c0:a8:38:fe STALE
192.168.56.12 dev eth0 lladdr 02:42:c0:a8:38:0c STALE
```

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

L'activation du routage sur la machine de l'attaquant est indispensable pour une attaque de l'homme du milieu (MITM) car elle permet à l'attaquant de rediriger le trafic, modifier les données en transit, écouter discrètement et réacheminer les réponses, ce qui est essentiel pour mener à bien l'attaque.

- 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, d'analyser et de manipuler le trafic entre la victime et la cible, tout en restant invisible pour la victime.

- Q13. Indiquer en quoi une attaque de type ARP Spoofing peut être utile ici au pirate.

Cette attaque peut être utile à un pirate dans le contexte d'une connexion SSH pour tromper la machine cliente en faisant croire qu'elle communique avec le mauvais serveur. Cette attaque permet aussi de fournir un accès non autorisé à des systèmes distants et permet d’intercepter des informations sensibles.

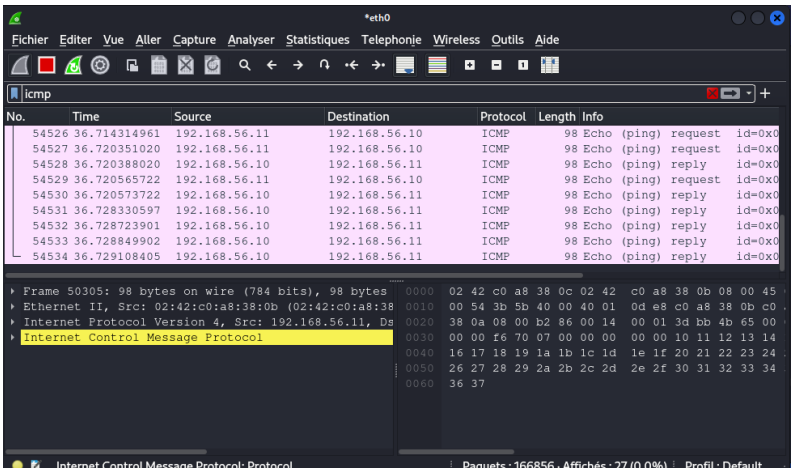
- Q14. Comparer les caches ARP du client et du serveur avec les associations notées précédemment lors de la question 10. Qu’en concluez-vous ?

Après cette manipulation, on peut voir que les adresses MAC sont toutes identiques. On peut en conclure que le kali se fait passer pour le serveur.

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

Une attaque ARP Spoofing implique qu'un attaquant envoie des paquets ARP falsifiés sur un réseau local pour faire correspondre des adresses IP légitimes avec sa propre adresse MAC, lui permettant de rediriger le trafic réseau destiné à d'autres machines vers sa propre machine.

- Q16. Envoyer une requête ping (icmp-écho) depuis le client vers le serveur (192.68.56.10). 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 ?



Sur cette capture on peut voir des requêtes ping réalisé entre la machine client (192.168.56.11) et la machine serveur (192.168.56.10). L'aperception de ses ping montre que l'attaque à bien été réalisé.

- 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 contient l'ensemble des informations collecter durant l'attaque SSH.

- Q18. Expliquer pourquoi ce message d'erreur apparaît.

Ce message d'erreur apparaît car l'attaque SSH est encore en cours. Ce message indique également que la clé d'hôte a possiblement été modifié.

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

Pour se connecter à nouveau, on doit ajouter la clé de connexion non modifier (la clé avant l'attaque) dans le fichier /home/etusio/.ssh/known_hosts et ensuite supprimer la fausse (clé durant l'attaque).

Partie 2 – Mise en place de contre-mesures

Dans le fichier de configuration du service SSH, on active l'authentification par clé de chiffrement et on indique le fichier où elles se trouvent :

```
etusio@clissh:~$ nano /var/tmp/ssh_dconfig
```

```
 GNU nano 7.2                                                    /var/tmp/ssh_dconfig
PubkeyAthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

Ensuite je redémarre le service :

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

Ensuite, l'utilisateur etusio doit générer sa clé publique et sa clé privée afin de pouvoir s'authentifier sur le serveur OpenSSH. Cela peut être réalisé à l'aide de la commande « ssh-keygen ».

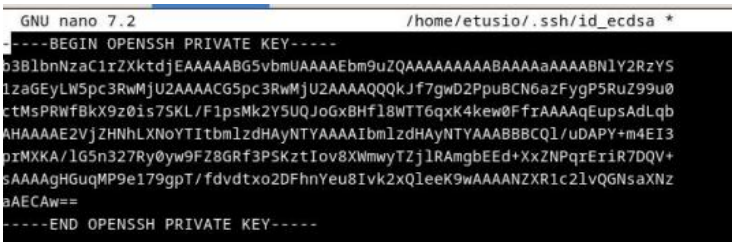
```
etusio@clissh:~$ ssh-keygen -b 256 -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/etusio/.ssh/id_ecdsa):
Created directory '/home/etusio/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/etusio/.ssh/id_ecdsa
Your public key has been saved in /home/etusio/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:xpqRnANafres053iNwShHD1BgQDAwbsNbglOQLqSWz0 etusio@clissh
The key's randomart image is:
+---[ECDSA 256]---+
|=+o... ++o       |
|oo    o +        |
|o . o. o o       |
|.* = oo+.        |
|O B E B S.       |
|.O . o O ..      |
|o     o.oo .     |
|       ..o =     |
|       .o.o .    |
+----[SHA256]-----+
```

Q1. Consultez le cache ARP de la machine cliente légitime avant de réaliser l'attaque et relevez l'adresse MAC de la passerelle :

L'utilisation de l'algorithme ECDSA pour la création de paires de clés a été privilégiée par rapport à RSA en raison de ses avantages, notamment une taille de clé réduite, une performance accrue, une sécurité adéquate et une utilisation optimale des ressources. Ces caractéristiques en font une option appropriée pour des environnements présentant des limitations de ressources et des exigences élevées en termes de performances.

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

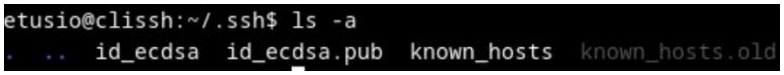
Il est essentiel d'établir une phrase de chiffrement pour sécuriser l'accès à la clé privée, ce qui contribue à prévenir les accès non autorisés, à garantir la confidentialité des données, à contrecarrer les attaques brute force et à conférer un contrôle personnel sur la clé privée. Cette mesure revêt une importance capitale. En exécutant la commande nano /home/etusio/.ssh/id_ecdsa, je suis en mesure d'inspecter ma clé privée.



J'exécute ensuite la commande nano /home/etusio/.ssh/id_ecdsa.pub pour voir ma clé publique.

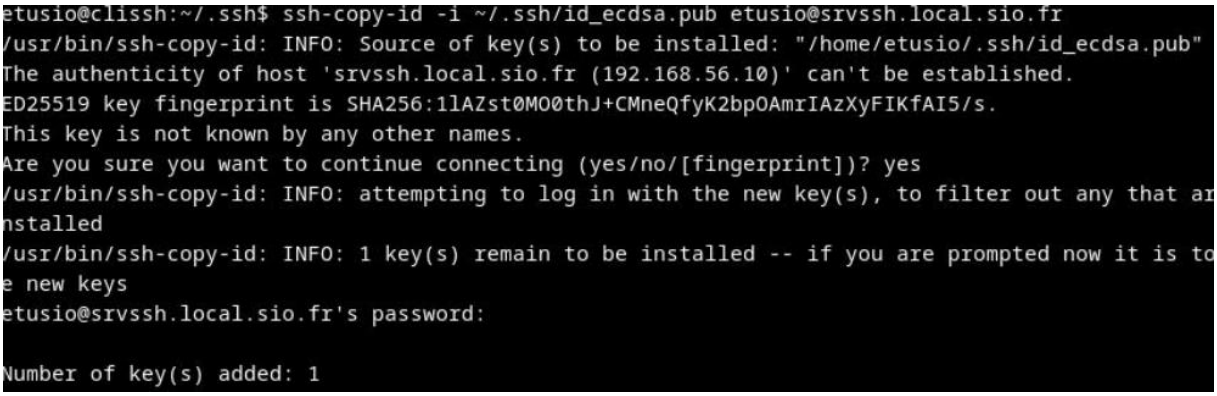
Q3. Listez le contenu du répertoire $HOME/.ssh/ puis afficher le contenu du fichier id_ecdsa.pub.

En utilisant la commande cd /home/etusio/.ssh/, je me déplace vers le répertoire approprié, puis avec la commande ls -a, j'affiche l'ensemble des fichiers présents dans le dossier /home/etusio/.ssh/.



À l'aide de la commande nano /home/etusio/.ssh/id_ecdsa.pub, il est possible d'inspecter la clé privée.

Je copie ensuite cette clé vers le serveur



En exécutant la commande nano /etc/ssh/sshd_config, je vais désactiver l'authentification par mot de passe afin de ne conserver que l'authentification par clés.

```
Serveur   Client
GNU nano 7.2                                    /etc/ssh/sshd_config *

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
```

Pour que les modifications que je viens d'apporter soient prises en compte, je relance le service SSH en utilisant la commande « sudo serveice ssh restart ».

<u>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 ?</u>

Modification des droits du fichier en 644 :

```
etusio@clissh:~/.ssh$ sudo chmod 644 id_ecdsa
```

Lors de la tentative de connexion au serveur, un message d'erreur signale que la clé privée n'est pas protégée avec les permissions actuelles, empêchant ainsi la connexion.

```
etusio@clissh:~$ ssh etusio@srvssh.local.sio.fr
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         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.
```

<u>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.</u>

```
etusio@srvssh:~$ ls -l ~/.ssh/authorized_keys
-rw------- 1 etusio etusio 175 10 nov.  16:12 /home/etusio/.ssh/authorized_keys
```

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

Suite au remplacement de la clé, l'impossibilité de se connecter est attribuée à un problème de permissions qui ne sécurise plus les clés privées du fichier authorized_keys.

```
etusio@srvssh:~$ chmod 666 ~/.ssh/authorized_keys
etusio@srvssh:~$ exit
déconnexion
Connection to srvssh.local.sio.fr closed.
etusio@clissh:~$ ssh etusio@srvssh.local.sio.fr
etusio@srvssh.local.sio.fr: Permission denied (publickey).
```

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 mot de passe d'une connexion par clé de chiffrement.

Les connexions par mot de passe s'authentifient au moyen d'une combinaison de caractères, tandis que les connexions par clé de chiffrement s'appuient sur l'utilisation d'une paire de clés (publique et privée) pour renforcer la sécurité.

Q8. Suite à la mise en place de cette authentification par clés de chiffrement, tenter à nouveau de simuler une attaque MITM entre le client et le serveur SSH. Quel résultat obtenez-vous ? Pourquoi ?

Après avoir mis en place cette méthode d'authentification, j'ai tenté à nouveau de simuler une attaque MITM entre le client et le serveur SSH, sans succès. Ensuite, j'ai supprimé l'intégralité du fichier known_hosts en utilisant la commande nano/home/etusio/.ssh/known_host.

Q9. Expliquer l'intérêt de cette démarche.

Valider manuellement la clé et l'enregistrer dans le DNS de l'entreprise permet de gagner considérablement du temps. Une fois validée, cette clé sera automatiquement reconnue par tous les postes de l'entreprise, simplifiant ainsi le processus d'authentification.

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

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

1) En utilisant la commande ls -l /etc/ssh/, on peut afficher la liste des fichiers ainsi que leurs permissions :

Les fichiers se terminant par "key" contiennent les clés privées. Il est notable qu'ils sont associés à l'utilisateur root, et que seul le compte root détient les droits d'accès sur ces fichiers.

```
etusio@clissh:~$ ls -l /etc/ssh
total 604
-rw-r--r-- 1 root root 573928  8 févr.  2023 moduli
-rw-r--r-- 1 root root   1650  8 févr.  2023 ssh_config
drwxr-xr-x 2 root root   4096  8 févr.  2023 ssh_config.d
-rw-r--r-- 1 root root   3223  8 févr.  2023 sshd_config
drwxr-xr-x 2 root root   4096  8 févr.  2023 sshd_config.d
-rw------- 1 root root    513  1 sept. 14:57 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    182  1 sept. 14:57 ssh_host_ecdsa_key.pub
-rw------- 1 root root    411  1 sept. 14:57 ssh_host_ed25519_key
-rw-r--r-- 1 root root    102  1 sept. 14:57 ssh_host_ed25519_key.pub
-rw------- 1 root root   2610  1 sept. 14:57 ssh_host_rsa_key
-rw-r--r-- 1 root root    574  1 sept. 14:57 ssh_host_rsa_key.pub
```

Pour vérifier le protocole par défaut, on peut utiliser la commande ssh etusio@192.168.56.10 -v. Cette commande affiche des informations détaillées sur la connexion, y compris le protocole utilisé.

2)

Par défaut, le protocole est déjà le 2. On peut le vérifier à l'aide de la commande ssh etusio@192.168.56.10 -v

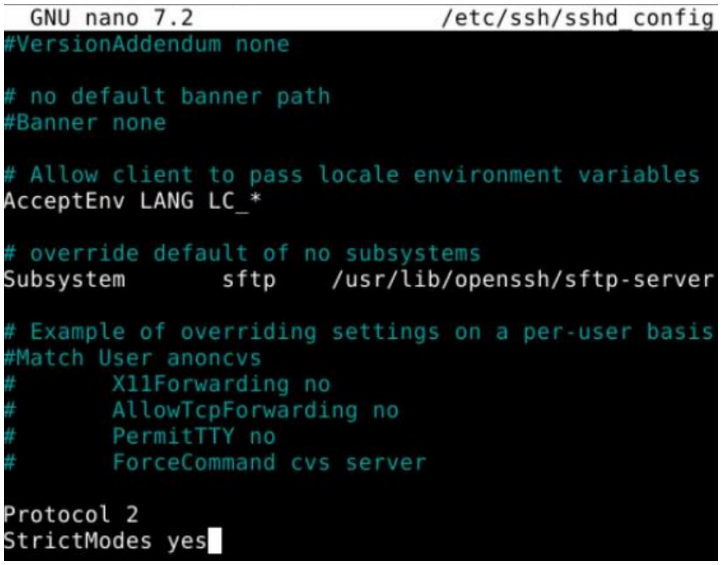
3)

Pour changer le port de 22 à 222 et enlever le symbole "#" qui est utilisé pour indiquer un commentaire, on peut éditer le fichier de configuration /etc/ssh/sshd_config.

```
  GNU nano 7.2                    /etc/ssh/sshd_config *

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

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

# The strategy used for options in the default sshd_co
# OpenSSH is to specify options with their default val
# possible, but leave them commented.  Uncommented opt
# default value.

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

Port 222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
```

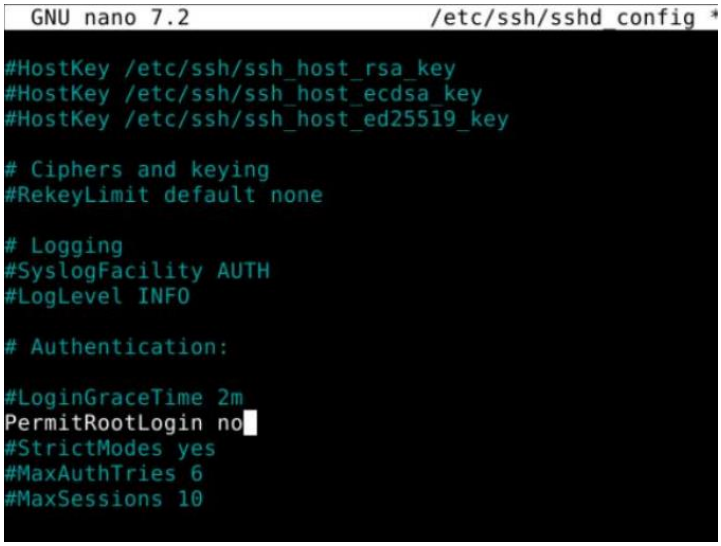
4)

On ajoute dans le fichier de configuration le paramètre StrictModes yes



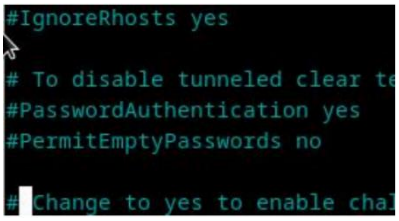
5)

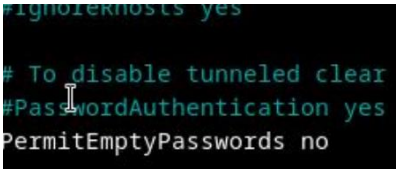
En configuration par défaut, l'accès SSH à l'utilisateur root est désactivé lors de l'utilisation du mot de passe, mais activé avec l'utilisation de clés publiques. Pour modifier cela dans le fichier, il suffit de supprimer le symbole # devant PermitRootLogin et de changer la valeur à no dans le fichier de configuration /etc/ssh/sshd_config.



7)

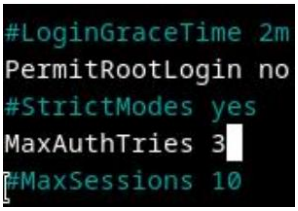
Le paramètre qui gère le blocage de la connexion aux utilisateurs n'ayant pas de mot de passe est PermitEmptyPasswords. Il doit être réglé sur no, ce qui est la configuration par défaut, mais le symbole # devant indique que c'est un commentaire. Par conséquent, nous devons supprimer le # pour activer ce paramètre dans le fichier de configuration /etc/ssh/sshd_config.





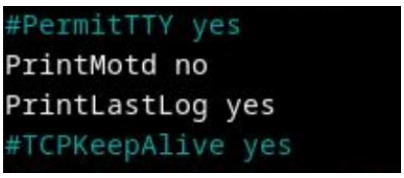
8)

Pour limiter le nombre de tentatives de connexion à 3, il est nécessaire de régler le paramètre MaxAuthTries sur 3 dans le fichier de configuration /etc/ssh/sshd_config.



9)

Le paramètre activé est PrintLastLog, qui est déjà configuré par défaut sur yes. J'ai simplement eu besoin de retirer le symbole # devant ce paramètre dans le fichier de configuration /etc/ssh/sshd_config.



10)

En modifiant la valeur de AllowUsers de ALL à etusio, on restreint l'autorisation d'accès uniquement à l'utilisateur "etusio". Cette modification doit être effectuée dans le fichier de configuration /etc/ssh/sshd_config.

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

L'état de l'art en cybersécurité représente le niveau le plus avancé des connaissances, des technologies et des pratiques actuelles visant à prévenir, détecter et contrer les menaces informatiques. Il garantit ainsi une protection efficace des systèmes, des réseaux et des données contre les attaques potentielles.

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

Le principe de Kerckhoffs stipule que la sécurité d'un système cryptographique ne doit pas dépendre du secret de son fonctionnement, mais uniquement de la confidentialité de la clé. En d'autres termes, la sûreté du système doit persister même si tous les détails de l'algorithme sont connus, mettant ainsi l'accent sur l'importance de la protection des clés cryptographiques.

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, il est pertinent de choisir des algorithmes cryptographiques connus et respectant l'état de l'art, car la sécurité d'un système ne doit pas dépendre du secret de l'algorithme lui-même. En optant pour des algorithmes largement étudiés, testés et acceptés par la communauté, la sécurité du système repose davantage sur la confidentialité de la clé, renforçant ainsi sa robustesse face aux attaques potentielles.