SSH

Table des matières

I. Introduction

SSH signifie *Secure Shell*. C'est un protocole qui permet de faire des connexions sécurisées (i.e. chiffrées) entre un serveur et un client SSH. Nous allons utiliser le programme OpenSSH, qui est la version libre du client et du serveur SSH.

Installer un serveur SSH permet aux utilisateurs d'accéder au système à distance, en rentrant leur login et leur mot de passe (ou avec un mécanisme de clefs). Cela signifie aussi qu'un pirate peut essayer d'avoir un compte sur le système (pour accéder à des fichiers sur le système ou pour utiliser le système comme une passerelle pour attaquer d'autres systèmes) en essayant plein de mots de passes différents pour un même login (il peut le faire de manière automatique en s'aidant d'un dictionnaire électronique). On appelle ça une attaque *en force brute*.

Il y a donc trois contraintes majeures pour garder un système sécurisé après avoir installé un serveur SSH :

- avoir un serveur SSH à jour
- Avec des mot de passe complexes
- surveiller les connexions en lisant régulièrement le fichier de log /var/log/auth.log.

Les mots de passes des utilisateurs sont stockés dans le fichier /etc/shadow :

```
apt install john
john /etc/shadow
```

Quand *john* a trouvé un mot de passe, il l'affiche avec le login associé.

II. Système de clés SSH

cryptographie asymétrique RSA ou DSA, chaque personne dispose d'un couple de clef : une clé publique et une clef privée. La clé publique peut être librement publiée tandis que la clef privée doit rester secrète.

Si Charlotte veut envoyer un message confidentiel à Evan, elle doit le chiffrer avec la clef publique de Evan et lui envoyer sur un canal qui n'est pas forcément sécurisé. Seul Evan pourra déchiffrer ce message en utilisant sa clef privée.

Cryptographie symétrique :

Si Sophie veut écrire confidentiellement a Loris, ils doivent tous les deux posséder la même clé secrète. Sophie chiffre son message non sécurisé et Loris peut le déchiffrer avec la clé secrète. Toute personne possédant la clé secrète peut aussi le déchiffrer.

Problème : Les algorithmes de cryptographie asymétrique sont très gourmand en ressources processeur … de plus comment faire pour échanger la clé secrète symétrique.

Dans le protocole SSL, qui est utilisé par SSH et par les navigateurs Web, la cryptographie asymétrique est utilisée au début de la communication pour que Sophie et Evan puissent s'échanger une clef secrète de manière sécurisée, puis la suite la communication est sécurisée grâce à la cryptographie symétrique en utilisant la clef secrète ainsi échangée.

L'établissement d'une connexion SSH :

Un serveur SSH dispose d'un couple de clefs RSA stocké dans le répertoire /etc/ssh/ et généré lors de l'installation du serveur.
Le fichier ssh_host_rsa_key contient la clef privée et a les permissions 600.
Le fichier ssh_host_rsa_key.pub contient la clef publique et a les permissions 644.

Ils ont ces permission car 600 permet de donner les droits de lecture et d'écriture seulement à l'utilisateur et 644 permet de donner les droits de lecture et d'écriture seulement à l'utilisateur ainsi que le droit de lecture aux groupes et aux fichiers.

étapes connexion SSH :

1. Le serveur envoie sa clé publique au client.
2. Le client génère sa clé secrète et l'envoie au serveur après l'avoir chiffré avec la clé publique du serveur. Le serveur déchiffre cette clé secrète grâce à sa propre clé secrète, ce qui prouve qu'il est le serveur.
3. Pour prouver qu'il est le serveur, celui-ci envoie un message standard avec sa clé secrète du client. Si le client retrouve le message standard avec sa clé secrète, c'est le vrai serveur.
4. Client et serveur ont la clé secrète, ils passent en symétrique et peuvent échanger login et mdp. Le canal sécurisé reste en place jusqu'à la déconnexion.

III. Installation

```
urpmi openssh-server
apt install openssh-server
```

On vérifie le démarrage du service

Le fichier de configuration du serveur SSH est /etc/ssh/sshd_config. À ne pas confondre avec le fichier /etc/ssh/ssh_config, qui est le fichier de configuration du client SSH.

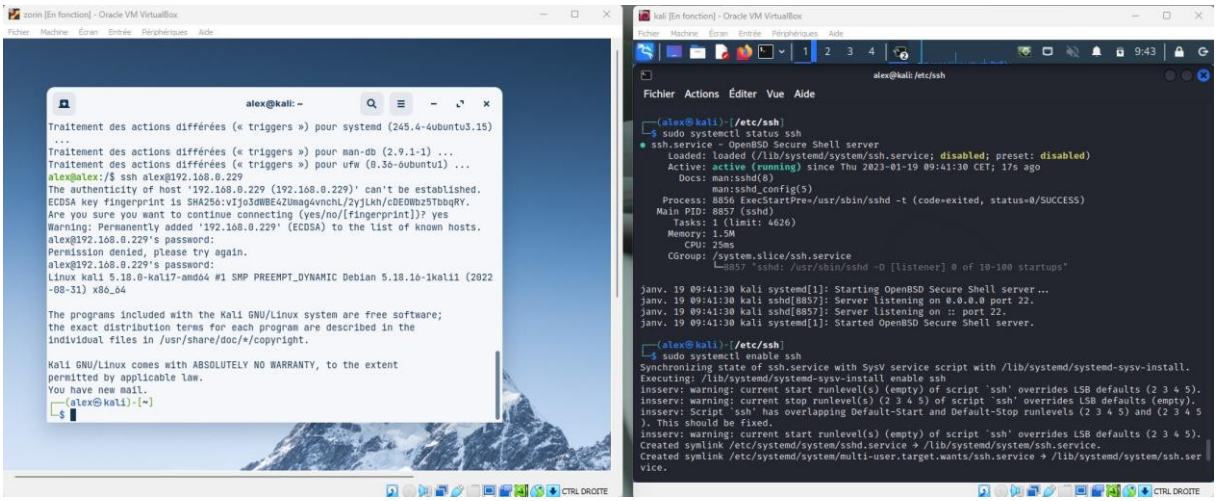
Port 22
PermitRootLogin yes/non
X11Forwarding yes/no

IV. Connexion

- Test de connexion

Pour se connecter il faut utiliser la commande :

```
ssh utilisateur@IPsrveur
```



On peut voir que je me suis connecter a ma VM Kali avec ma VM Zorin, sur kali j'ai activer le service au démarrage avec la commande :

```
sudo systemctl enable ssh
```

Je vais modifier les paramètres pour limiter le nombre de tentative de connexion à 5, interdire les connexion root, modifier le port mais avant tout cela je fais un copie du fichier de configuration sshd_config que je nomme default_sshd_config

Limiter le nombre de tentative de connexion à 5 :

Dans le fichier de configuration modifie les options LoginGraceTime et MaxAuthTries en leurs donnant les valeurs suivante 2m et 5. Ces options vont permettre respectivement d'attendre deux minutes avant de refuser un connexion et d'autoriser seulement 5 tentative de connexion, cela nous permet d'éviter les tentative de brute force. Je modifie également le port dans la ligne Port 22 par le 3452, pour vérifier si le port est utiliser on peut utiliser la commande suivante :

```
Sudo netstat -tulnp | grep 3452
```

Si rien n'est retourner dans le terminal ce qu'aucun service ne se sert du port

```
┌──(alex㉿kali)-[/etc/ssh]
└─$ sudo netstat -tulnp | grep 3452

┌──(alex㉿kali)-[/etc/ssh]
└─$ sudo nano sshd_config

┌──(alex㉿kali)-[/etc/ssh]
└─$ sudo systemctl restart ssh

┌──(alex㉿kali)-[/etc/ssh]
└─$ sudo netstat -tulnp | grep 3452
tcp        0      0 0.0.0.0:3452            0.0.0.0:*               LISTEN      15855/sshd: /usr/sb
tcp6       0      0 :::3452                 :::*                    LISTEN      15855/sshd: /usr/sb
```

On voit ici qu'après la commande rien n'est retourner donc aucun service n'utilise ce port puis je modifie le fichier sshd_config en changeant le port du ssh.

En retapant la commande cette fois on voit que le service ssh utilise ce port .

- Test de scp

Pour transférer un fichier de mon client au serveur j'utilise la commande suivante sans me connecter en ssh :

Scp -P 3452 /zorin.txt alex@192.168.0.229/home/alex/zorin.txt

```
alex@alex:/$ scp -P 3452 /zorin.txt alex@192.168.0.229:/home/alex/zorin.txt
alex@192.168.0.229's password:
zorin.txt                                     100%    0     0.0KB/s   00:00
```

```
┌──(alex㉿kali)-[~]
└─$ ls
Bureau              Maltego.v4.3.1.deb      Public                  test1.txt
Documents           Modèles                 sherlock                Vidéos
Images              Musique                 'sinkcanned61 (1).ovpn' wifiphisher
'info total.txt'    PEASS-ng-20220918.zip   sinkcanned61.ovpn       ZAP_2_11_1_unix.sh
kalitorify          peass.rb                Téléchargements         zorin.txt
```

On voit que le fichier à bien été téléchargé sur le serveur.

Maintenant on va transférer un fichier du serveur au client pour cela on utilise la commande suivante depuis le client :

Sudo scp -P 3452 alex@192.168.0.229:/home/alex/test1.txt .

On constate que en faisant ls le fichier test1.txt et apparue en vert donc le téléchargement a réussi.

- Test de sftp

Le sftp est la version sécuriser de FTP les commande et le mode de connexion sont les même :

```
Sudo sftp -o Port=3452 alex@192.168.0.229
```

```
alex@alex:/$ sudo sftp -o Port=3452 alex@192.168.0.229
alex@192.168.0.229's password:
Connected to 192.168.0.229.
sftp> cd Bureau
sftp> ls
Nmap-Nessus-Cheat-Sheet.pdf            projets
sinkcanned61 (2).ovpn                  sinkcanned61.ovpn
testcommande
sftp> get Nmap-Nessus-Cheat-Sheet.pdf
Fetching /home/alex/Bureau/Nmap-Nessus-Cheat-Sheet.pdf to Nmap-Nessus-Cheat-Sheet.pdf
/home/alex/Bureau/Nmap-Nessus-Cheat-Sheet.pdf 100%   63KB   4.3MB/s   00:00
sftp> exit
alex@alex:/$ ls
bin    home    lost+found                  proc  srv        usr
boot   lib     media                       root  swapfile   var
cdrom  lib32   mnt                         run   sys        zorin.txt
dev    lib64   Nmap-Nessus-Cheat-Sheet.pdf  sbin  test1.txt
etc    libx32_ opt                         snap  tmp
```

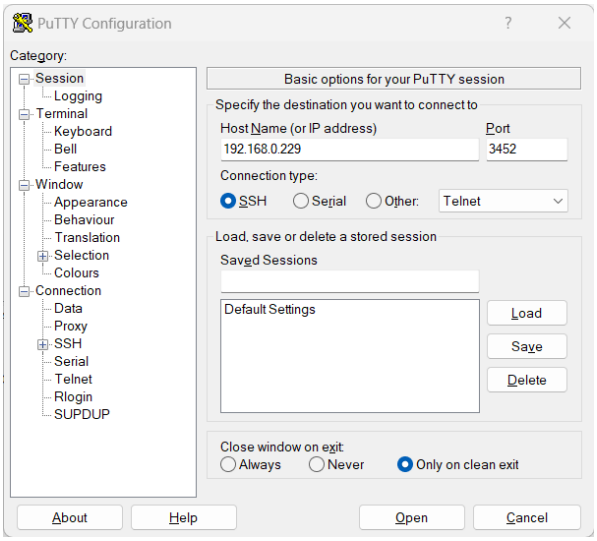
J'ai utilisé la commande get pour télécharger un fichier moais il en existe plein d'autre

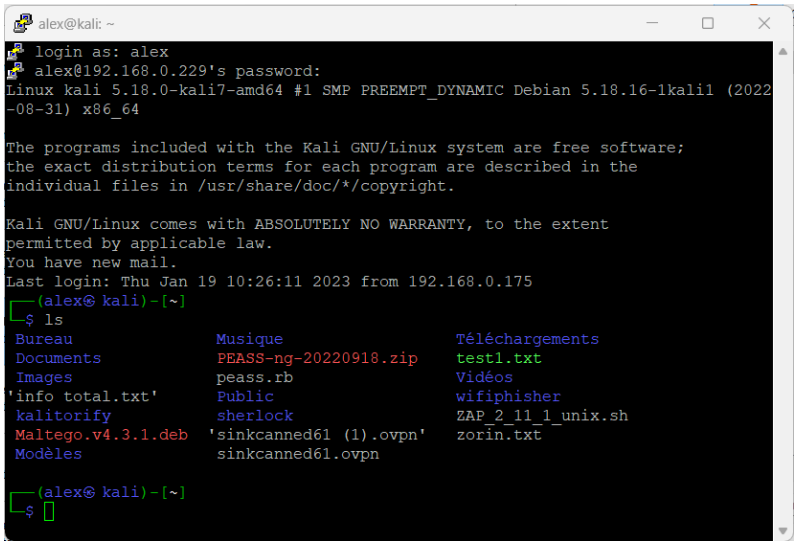
- help : affiche la liste des commandes disponibles,
- help commande : affiche l'aide de la commande,
- ls : liste le contenu du répertoire distant,
- ls -la : liste le contenu du répertoire distant avec les fichiers cachés et les permissions,
- cd répertoire : change de répertoire distant,
- lcd répertoire : change de répertoire local,
- get fichier : télécharge le fichier,
- get *.img : télécharge tous les fichiers avec l'extension *img*,
- mirror répertoire : télécharge le répertoire,

- put fichier : dépose le fichier,
- put test* : dépose tous les fichiers dont le nom commence par *test*,
- exit : met fin à la connexion.

- Test de putty, mobaXtern, cmd

PuTTY est un client SSH (Secure Shell) pour Windows. Il permet aux utilisateurs de se connecter à des serveurs distants via une interface graphique utilisateur (GUI) pour exécuter des commandes et transférer des fichiers. Il prend également en charge les connexions Telnet et Rlogin. Il est utilisé pour accéder à des serveurs Linux / Unix à partir d'un ordinateur Windows.





IL existe beaucoup d'autre client ssh comme le cmd, PowerShell, mobaXtern , Kitty, Winscp.

- Faire un tunnel SSH :

Faire un tunnel SSH est un moyen simple de chiffrer n'importe quelle communication TCP entre votre machine et une machine sur laquelle vous avez un accès SSH.

Pour activer le tunneling il faut se rendre dans le fichier /etc/ssh/sshd_conf et mettre yes a la ligne PermitTunnel.

Puis sur le client mofifier le fichier /etc/ssh/ssh_conf et mettre yes a la ligne Tunnel.
Il nous reste plus qu'a rentrer la commande suivante dans le terminal du client :

```
ssh -L -N 2012:192.168.0.212:80 alex@192.168.0.212
```

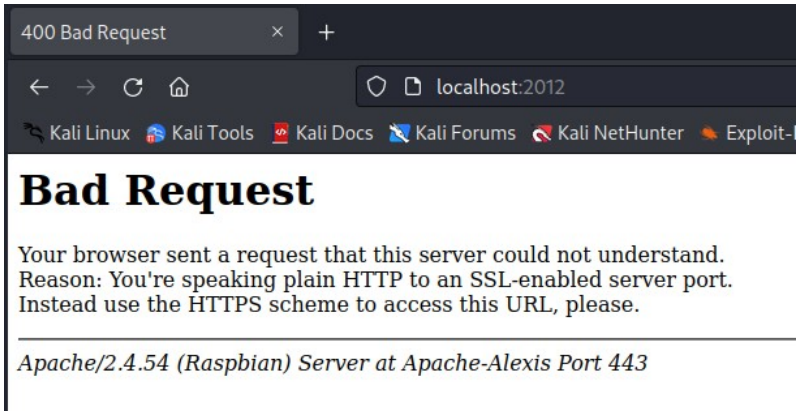
```
(alex㉿kali)-[~]
$ ssh  -N -L  2012:192.168.0.212:443 alex@192.168.0.212
alex@192.168.0.212's password:
Tunnel device open failed.
Could not request tunnel forwarding.
```

Ensuite dans un autre terminal je vérifie si le port 2012 est ouvert :

```
┌──(alex㉿kali)-[~]
└─$ sudo netstat -tulnp | grep 2012
tcp        0      0 127.0.0.1:2012          0.0.0.0:*               LISTEN      59523/ssh

tcp6       0      0 ::1:2012                :::*                    LISTEN      59523/ssh
```

Il nous reste plus qu'à aller sur un navigateur et taper localhost :2012



On peut voir que l'on a accès au serveur ; le Bad Request est la car j'ai pris une VM qui avait un serveur apache mal configurer je l'ai donc refait sur Ubuntu et voilà le résultat.