

# TP SSH partiel 23/04/2024

Ivan KRIVOKUCA , Abdel-malik FOFANA (Groupe 12)

## 1) Création client-serveur SSH

### 1.1) Serveur

### 1.2) Client

## 2) Explication du fonctionnement du serveur SSH

## 1) Création client-serveur SSH

### 1.1) Serveur

Installation du serveur (machine sous **Debian**) :

```
sudo apt install openssh-server
```

Pour lancer le serveur `systemctl start sshd` , on peut aussi activer le service `systemctl enable ssh.service` pour éviter de le lancer à chaque démarrage de l'ordinateur.

```
root@debian:/home/debian# systemctl start sshd
root@debian:/home/debian# systemctl enable ssh.service
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
```

On part du principe que l'utilisateur qui veut se connecter est déjà présent sur la machine, sinon on effectue sur le serveur : `adduser NOM` . Ici nous ajouter l'utilisateur "*maliki*".

Le fichier de configuration se trouve dans : `/etc/ssh/sshd_config`

```
#Restriction d'utilisateur/groupe
AllowUsers user1 user2
AllowGroups group1

Port 22    # Changer le port où écoute SSH
PermitRootLogin no # Pour pas que root se connecte

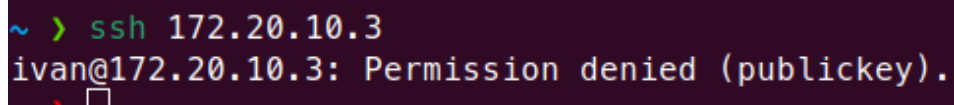
#Si on veut enlever la connexion par mot de passe
```

```
PasswordAuthentication no
#Il faut mettre ça aussi
PubkeyAuthentication yes

X11Forwarding yes # Pour avoir l'affichage X
```

Ici, pour cet exemple, on a mis les paramètres (ne pas oublier de restart le service pour appliquer les changements)

```
PasswordAuthentication no
PubkeyAuthentication yes
```



```
~ > ssh 172.20.10.3
ivan@172.20.10.3: Permission denied (publickey).
```

Ici, l'utilisateur *ivan* n'a pas transmis sa clé publique au serveur et vu que la connexion par mot de passe a été désactivée, il lui est maintenant impossible de se connecter.

## 1.2) Client

Ici, l'IP du serveur est : **172.20.10.3**

Création d'une clé client avec `ssh-keygen`

```
(maliki@maliki)-[~]
$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/maliki/.ssh/id_ed25519):
Created directory '/home/maliki/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/maliki/.ssh/id_ed25519
Your public key has been saved in /home/maliki/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:BjYeLkkW0XHG7zG6/kUPbyXwTtjYkOd0u7rPXc2fDl0 maliki@maliki
The key's randomart image is:
+--[ED25519 256]--+
|      oo.oo      |
|     ..o. .      |
|    o = . + .    |
|   o = + + @     |
|  o o So o= O E  |
| . . . . O.=o    |
|  . . .B.+       |
| . . o.o+        |
| .... o+=oo      |
+-----[SHA256]-----+
```

`ssh-keygen` génère une paire de clés pour l'identification.

Ainsi, on créera une clé privée et une clé publique. Ici, on utilisera l'algorithme `ed25519` (celui par défaut).

On peut très bien le changer en utilisant l'option `-t` suivant de l'algorithme. Par exemple : `-t rsa -b 4096` créer une jeu de clé privée/publique utilisant l'algorithme RSA avec une taille de 4096 bits.

On envoie la clé publique au serveur, ce qui permettra au client de ne pas à rentrer son mot de passe à chaque connexion.

```
ssh-copy-id maliki@172.20.10.3
```

```
(maliki@maliki)-[~/cyber m1/reseau web/projet web/lamp]
$ ssh-copy-id maliki@172.20.10.3
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
maliki@172.20.10.3's password:

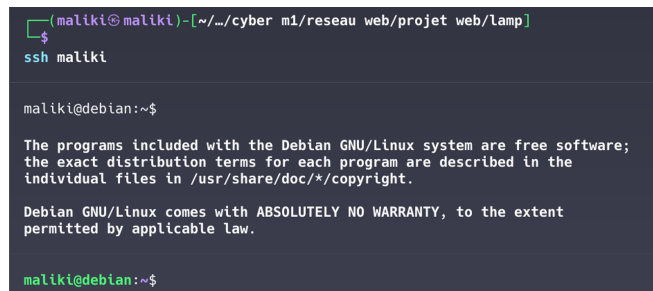
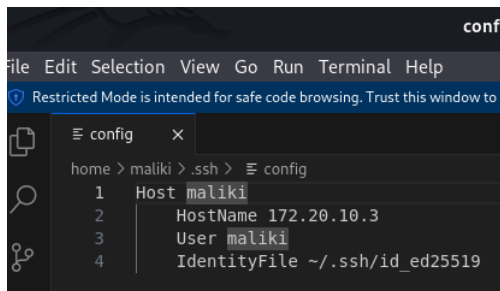
Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'maliki@172.20.10.3'"
and check to make sure that only the key(s) you wanted were added.
```

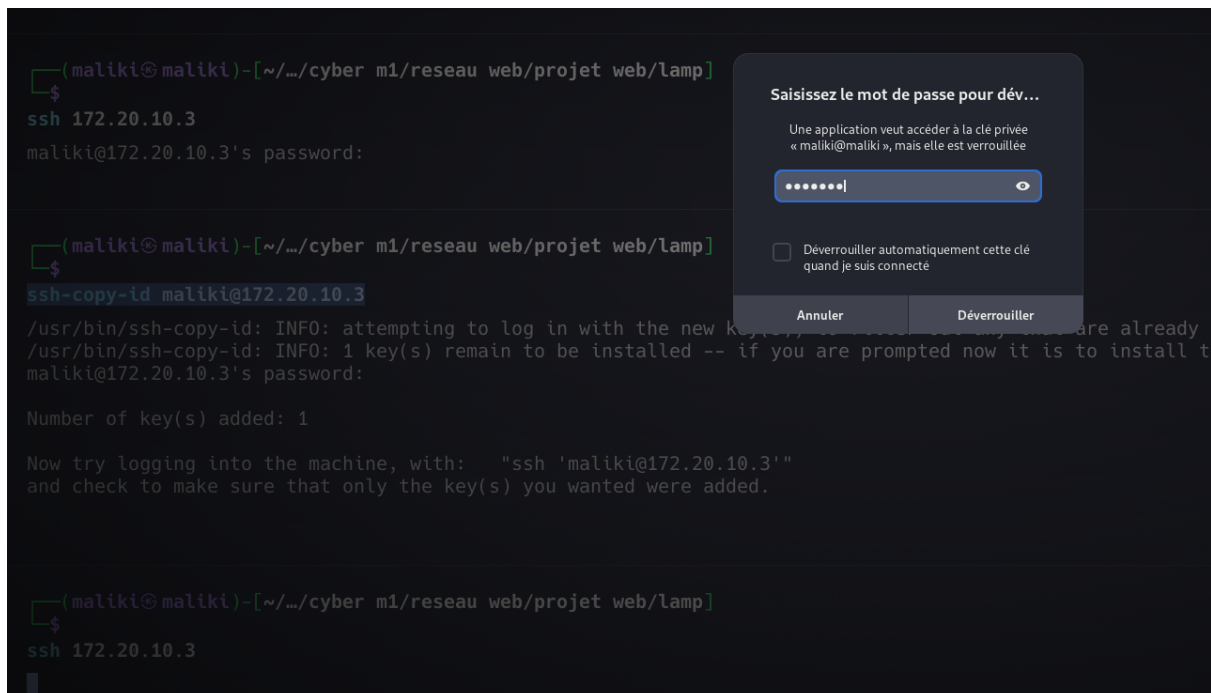
Le serveur va mettre cette clé dans le fichier `~/.ssh/authorized_keys`

```
maliki@debian:~/.ssh$  
cat authorized_keys  
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMge85MdadYmwctv5nsg9uvA0UBf6HsYhRm6gHk7VLMY maliki@maliki
```

Le fichier `~/.ssh/config` permet d'ajouter des configuration SSH, ici on utilisera pour se connecter : `ssh maliki` (au lieu de `ssh maliki@172.20.10.3`)



Lorsque l'on se connecte, on doit entrer le mot de passe si celui-ci a été défini lors de la création de la clé.



Maintenant nous avons plus besoin de mettre notre mot de passe lorsque nous nous reconnectons comme demandé.

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

```
maliki@debian:~$
exit
exit
Connection to 172.20.10.3 closed.
```

```
(maliki@maliki)-[~/.../cyber m1/reseau web/projet web/lamp]
$
ssh 172.20.10.3
```

```
maliki@debian:~$

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.

maliki@debian:~$
```

## 2) Explication du fonctionnement du serveur SSH

Depuis le serveur :

```
ps aux |grep "ssh"
```

Voici les services SSH en utilisation :

```
maliki@debian:~/.ssh$
ps aux |grep "ssh"
debian 1696 0.0 0.1 88372 5576 ? Ssl 14:09 0:00 /usr/libexec/gcr-ssh-agent /run/user/1000/gcr
debian 1698 0.0 0.1 7800 5100 ? Ss 14:09 0:00 ssh-agent -D -a /run/user/1000/openssh_agent
root 5418 0.0 0.3 15412 9460 ? Ss 14:10 0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
root 7156 0.0 0.3 17760 11068 ? Ss 14:25 0:00 sshd: maliki [priv]
maliki 7172 0.0 0.2 18024 7164 ? S 14:25 0:00 sshd: maliki@pts/2
maliki 7786 0.0 0.0 6352 2160 pts/2 S+ 14:37 0:00 grep ssh
```

- **/usr/libexec/gcr-ssh-agent** : Gère les clés privées SSH de manière sécurisée.
- **ssh-agent** : Stocke les clés privées SSH en mémoire pendant la session.
- **sshd** : Écoute les connexions SSH entrantes et gère l'authentification des utilisateurs.
- **sshd: maliki [priv]** : Gère les connexions SSH pour l'utilisateur "maliki".

- **sshd: maliki@pts/2** : Indique qu'un utilisateur "maliki" est connecté via SSH sur un terminal.

On peut aussi utiliser la commande **who** (côté serveur) pour savoir quels utilisateurs sont connectés

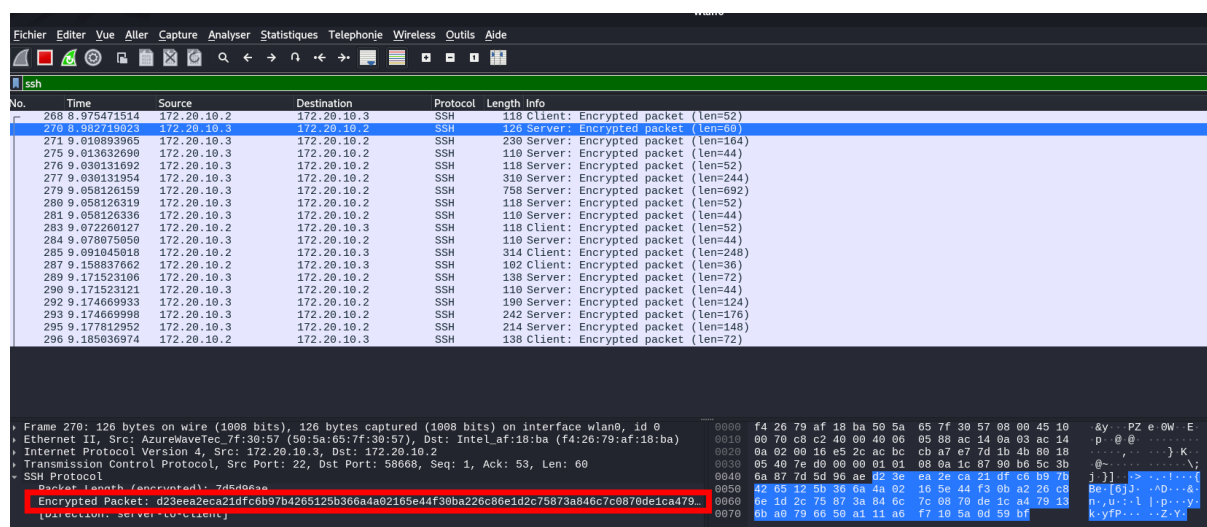
```
root@debian:/home/debian# who
debian    tty2          2024-04-23 14:09 (tty2)
debian    pts/1         2024-04-23 14:09
maliki    pts/2         2024-04-23 14:25 (172.20.10.2)
```

On peut aussi utiliser **last** qui nous permet de savoir la dernière connexion

```
root@debian:/home/debian# last
maliki    pts/2         172.20.10.2    Tue Apr 23 15:31    still logged in
maliki    pts/2         172.20.10.2    Tue Apr 23 15:17 - 15:21 (00:03)
```

Le but de SSH est de chiffrer les données envoyées entre le client et le serveur SSH.

On peut voir dans ce screen SSH que effectivement les paquets échangés entre le client et le serveur sont bien chiffrés



Il est important de mettre un mot de passe sécurisé pour se connecter au serveur SSH.

En effet avec un outil aussi simple que

**hydra** on peut hacker le mot de passe ssh en quelques secondes car le mot de passe est faible (ici "azerty")

```
(maliki@maliki)-[~/cyber m1/reseau web/projet web/lamp]
$ hydra -l ivan -P /usr/share/wordlists/rockyou2.txt 172.20.10.3 -t 5 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-23 15:03:22
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
ls
[DATA] max 4 tasks per 1 server, overall 4 tasks, 4 login tries (l:1/p:4), ~1 try per task
[DATA] attacking ssh://172.20.10.3:22/
[22][ssh] host: 172.20.10.3 login: ivan password: azerty
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-23 15:03:36
```

Pour “contrer” cela, on peut désactivé la connexion via mot de passe, changer le port ou bien configurer *iptables* ou alors ajouter *MaxAuthTries 3* dans le fichier de configuration qui coupera la connexion après 3 connexions.