TP B2 : Admin à distance : SSH

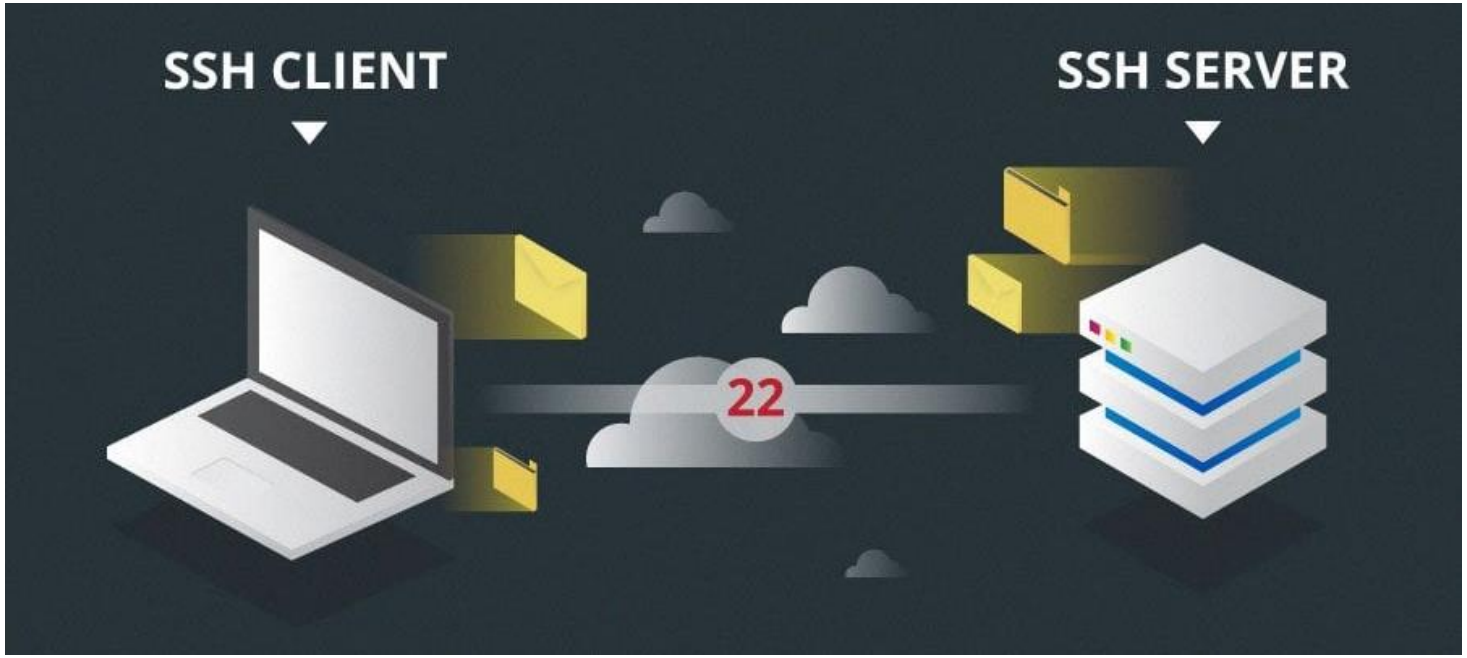
HEGO Maxence

SSH :

Dans ce TP on va mettre en place un serveur SSH sur une VM debian.

On utilise également une VM Debian qui sera utilisé en tant que client.

Le serveur SSH est à la fois un programme et un protocole de communication. Il sert à pouvoir se connecter et administrer un serveur à distance.



Installation serveur SSH :

On commence par installer le serveur SSH sur la VM debian correspondante.

Après avoir mis à jour les paquets, on installe le serveur SSH avec la commande : **apt install openssh-server**

```
root@debiansio:~# apt update
```

```
root@debiansio:~# apt install openssh-server
```

On peut maintenant vérifier que le serveur ssh est installé avec la commande : **which ssh**

Le serveur est bien installé, il est dans le dossier /usr/bin/ssh

```
root@debiansio:~# which ssh
/usr/bin/ssh
```

Configuration initiale : Users

Par défaut on ne peut pas se connecter en tant que Root par le SSH.

Afin de se connecter une première fois au serveur on commence par créer les utilisateurs.

On veut créer trois utilisateurs : User1, User2 et User3.

On créer les utilisateurs avec la commande : **adduser** *nomuser* puis on définit leur mot de passe.

```
root@debiansio:~# adduser user1
Ajout de l'utilisateur « user1 » ...
Ajout du nouveau groupe « user1 » (1001) ...
Ajout du nouvel utilisateur « user1 » (1001) avec le groupe « user1 » (1001) ...
Création du répertoire personnel « /home/user1 » ...
Copie des fichiers depuis « /etc/skel » ...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour user1
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []:
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Cette information est-elle correcte ? [O/n]o
Ajout du nouvel utilisateur « user1 » aux groupes supplémentaires « users » ...
Ajout de l'utilisateur « user1 » au groupe « users » ...
```

Configuration initiale : Mot de passe

Le mot de passe que nous avons défini pour les utilisateurs ne respecte pas les exigences en termes de sécurité.

On va donc le changer.

On peut également changer les mots de passe des utilisateurs avec la commande **chpasswd**.

```
root@debiansio:~# chpasswd
user3:Sio%2025
```

On entre ensuite le **login:nouveaumdp**

On sauvegarde et on quitte l'interface de changement avec la commande **crt D**

Configuration initiale : Groupes

Pour la suite du TP nous aurons également besoin de créer des groupes.

Ces groupes permettrons de filtrer la connexion au serveur aux seuls utilisateurs du groupe SSH.

On crée deux groupes : etudiants et ssh

On utilise la commande : groupadd nomdugroupe

```
root@debiansio:~# groupadd etudiants
root@debiansio:~# groupadd ssh
```

Pour attribuer les utilisateurs dans leurs groupes on utilise la commande : **usermod –a –G** *nomdugroupe nomdel'user*

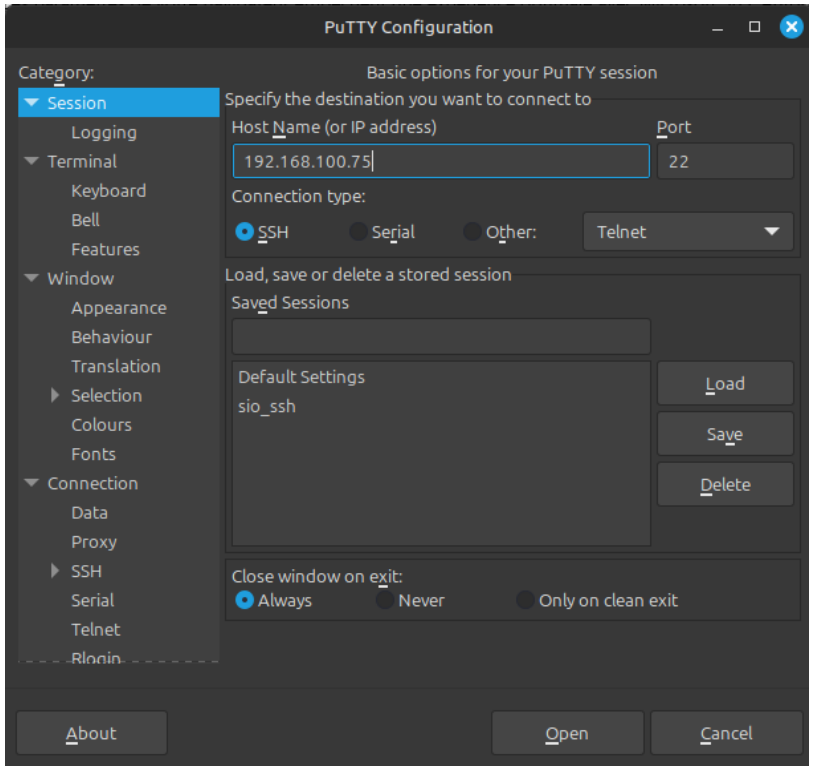
```
root@debiansio:~# usermod -a -G ssh user1
root@debiansio:~# usermod -a -G ssh user2
root@debiansio:~# usermod -a -G etudiants user1
root@debiansio:~# usermod -a -G etudiants user3
```

Le groupe SSH contient les user1 et user2, le groupe étudiants contient les user1 et user3

Connexion au serveur :

On peut maintenant essaye de se connecter au serveur.

On utilise pour cela le client Putty



On se connecte au serveur en utilisant Putty. On y entre l'adresse IP du serveur debian.

On se connecte en entrant le login et le mot de passe d'un utilisateur. Ici on utilise login : user1

Connexion au serveur : teste des droits

Avec le nouvel utilisateur on vérifie si on peut modifier le fichier sshd_config

```
user10@debiansio:~$ nano /etc/ssh/sshd_config
```

On essaie de modifier le fichier avec la commande : **nano /etc/ssh/sshd_config**

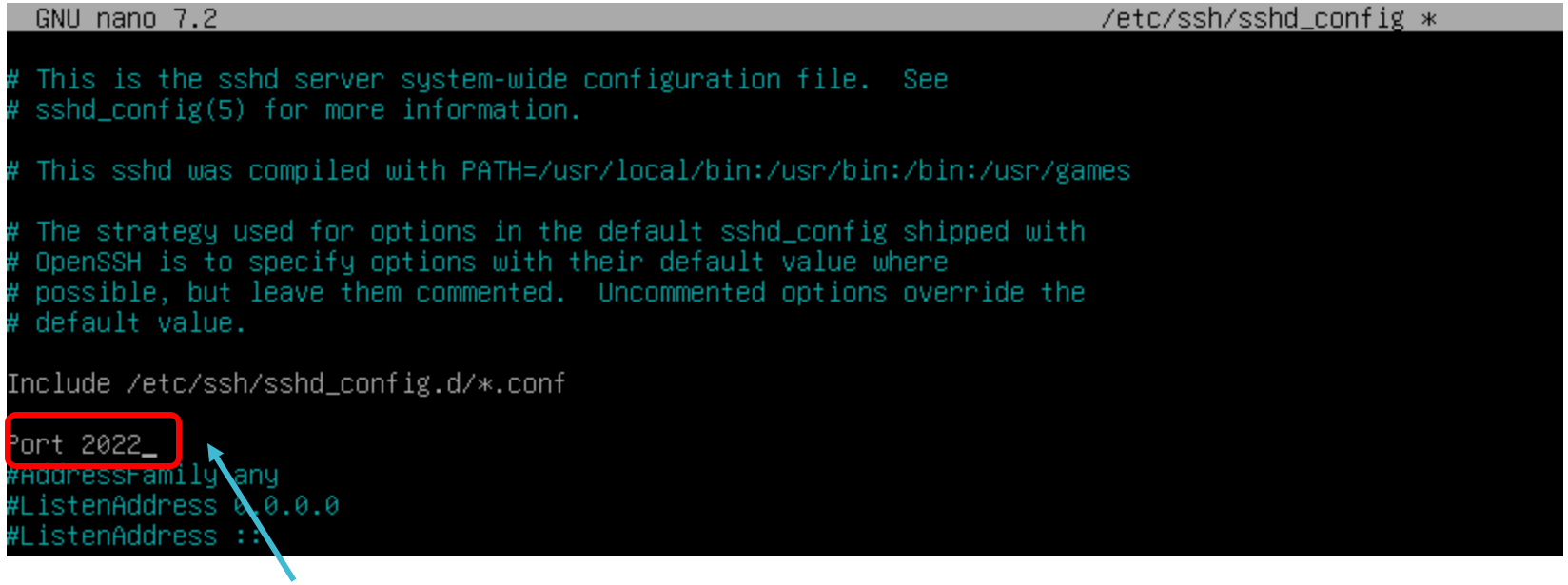
```
 GNU nano 7.2                    /etc/ssh/sshd_config

# 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
    [ Le fichier « /etc/ssh/sshd_config » n'est pas accessible en écriture ]
^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^T Exécuter   ^C Emplacement
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^J Justifier  ^/ Aller ligne
```

On peut afficher le dossier mais on n'a pas les permissions pour le modifier.

Changement du port d'écoute :

On veut modifier le port d'écoute du serveur.

Pour cela on se connecte sur le serveur en mode super utilisateur et on modifie le fichier **sshd_config**



On décommente la ligne et on remplace **le port 22** par le numéro du port que l'on veut utiliser

On peut se reconnecter au serveur immédiatement car les changements ne sont pas appliqués.

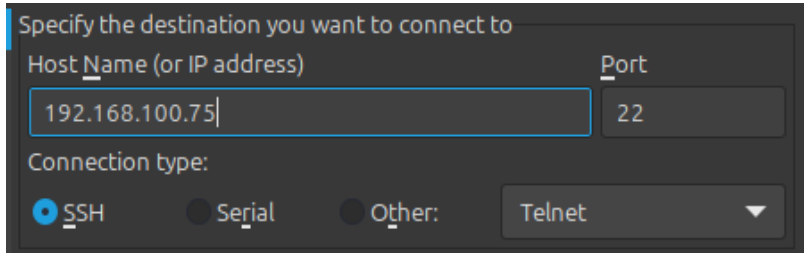
Pour faire appliquer les changements on redémarre SSH avec la commande **systemctl restart ssh**

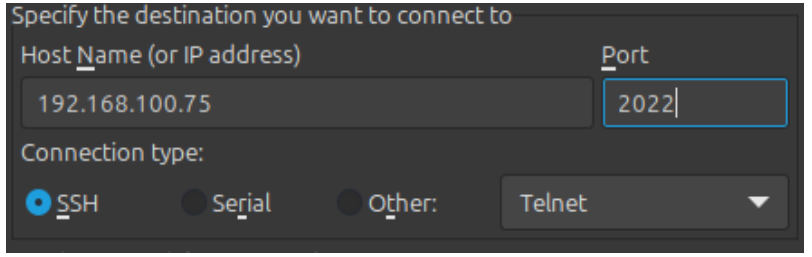
```
root@debiansio:~# systemctl  restart ssh
```

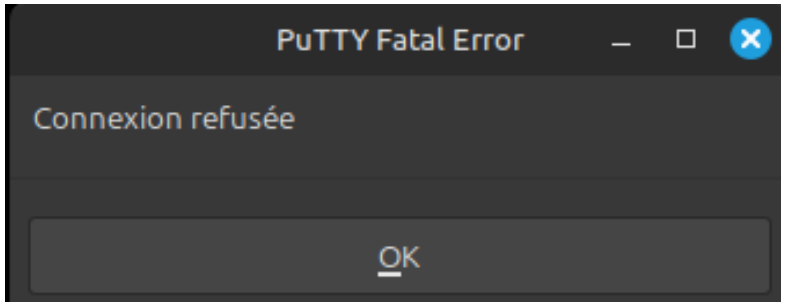
Changement du port d'écoute :

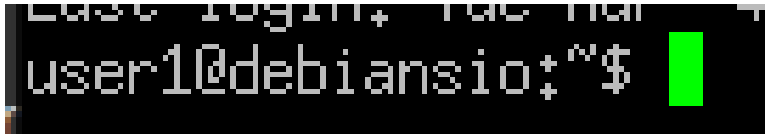
Après avoir redémarrer le service on essaie de se connecter au serveur.

Pour le reste du TP on repassera sur le port 22.









La connexion sur le **port 22** est refusé

On se connecte au serveur sur **le port 2022** conformément à la modification du fichier **sshd_config**

Changer le port d'écoute permet d'améliorer la sécurité. En effet le **port 22** est le **port par défaut** d'un service **SSH**, en le changeant on rajoute une couche de complexité pour les attaquants.

Configuration root login :

Dans le fichier configuration il existe l'option PermitRootLogin. Voyons l'utilité de cette commande.

La condition **permit root login autorise ou non un utilisateur a accéder au mode root** depuis le SSH.

Les trois valeurs possibles sont :

- **no** : pas de connexion root par SSH
- **yes** : connexion possible mais avec mdp
- **without-password** : connexion au SSH en root et sans mdp mais en utilisant une clé SSH

Il est recommandé de **laisser cette condition sur non,** en effet si un hacker parvient à se connecter en root au SSH, notamment par brute force, alors il a les privilèges absolus sur le système.

```
# Authentication:

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

Configuration password :

Une autre option est PermitEmptyPassword. Voyons à quoi sert cette option.

La ligne **PermitEmptyPassword** défini si un utilisateur qui n'a pas défini de mot de passe peut **se connecter** au serveur via **SSH.**

Par défaut cette option est sur **non** en raison des importants risques pour la sécurité.

Cette commande est différente de **PermitRootLogin without-password** qui ne s'occupe que de l'identification du super utilisateur et **demande une clé SSH** pour l'authentification.

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

Gérer les clés d'authentification :

Afin d'améliorer la sécurité de la connexion au serveur on va remplacer les mdp par des clés DSA.

Les utilisateurs que nous avons créés précédemment vont nous permettre de vérifier que l'on peut accepter ou non des utilisateurs donnés à un serveur SSH.

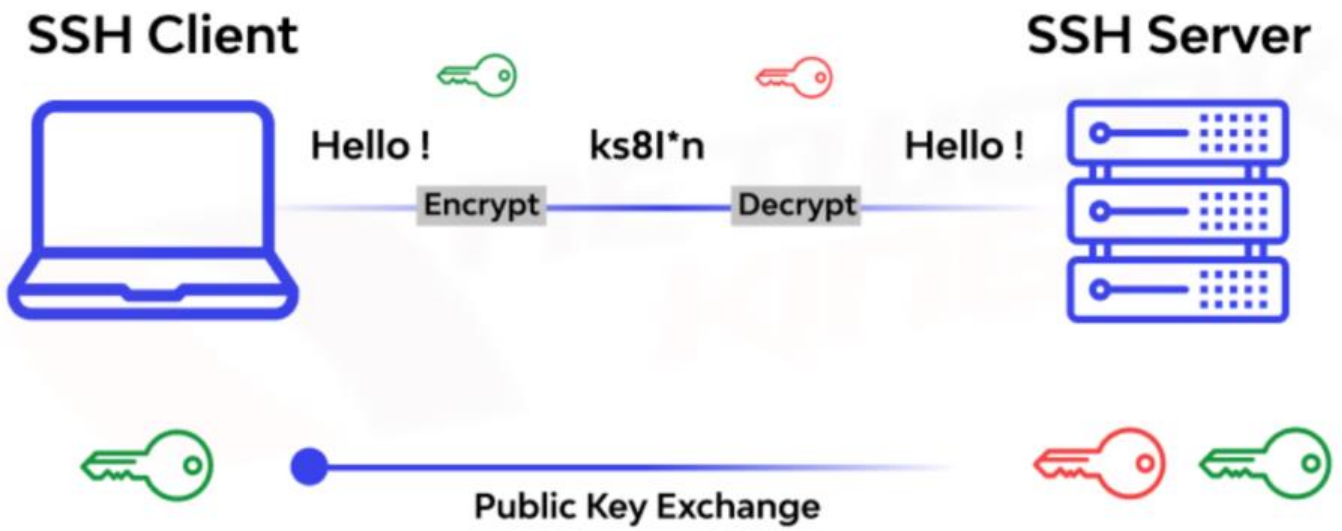
Seuls les utilisateurs appartenant au groupe SSH pourront se connecter au serveur.

Une authentification en SSH est plus sur car on ne transmet pas les mdp sur le réseau en utilisant une identification par clé DSA. Un hacker ne peut pas intercepter les communications et trouver le mot de passe.

SSH explication :

Quelques informations complémentaires sur le protocole SSH

SSH (Secure Shell) est un protocole qui permet de se connecter à un serveur distant de manière sécurisée. Il établit une connexion chiffrée entre le client et le serveur, négocie des algorithmes de chiffrement, authentifie l'utilisateur via mot de passe ou clés, et transfère les données de manière sécurisée, garantissant ainsi la confidentialité et l'intégrité des informations échangées.



Création clés : Préparation VM cliente

Avant de pouvoir créer les clés on prépare la machine cliente.

On créer les utilisateurs, les groupes et on attribues les users aux groupes comme on l'a fait pour la VM server.

```
root@debiansio:~# adduser user3
Ajout de l'utilisateur « user3 » ...
Ajout du nouveau groupe « user3 » (1003) ...
Ajout du nouvel utilisateur « user3 » (1003) avec le groupe « user3 » (1003) ...
Création du répertoire personnel « /home/user3 » ...
Copie des fichiers depuis « /etc/skel » ...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour user3
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []:
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Cette information est-elle correcte ? [O/n]o
Ajout du nouvel utilisateur « user3 » aux groupes supplémentaires « users » ...
Ajout de l'utilisateur « user3 » au groupe « users » ...
root@debiansio:~# groupadd etudiants
root@debiansio:~# groupadd ssh
root@debiansio:~# usermod -a -G ssh user1
root@debiansio:~# usermod -a -G ssh user2
root@debiansio:~# usermod -a -G etudiants user1
root@debiansio:~# usermod -a -G etudiants user3
```

Création des répertoires pour échange des clés :

On va maintenant créer les répertoires dans lesquelles seront stockées les clés DSA utilisé pour l'authentification.

Sur le serveur :

On se place dans le répertoire /home et on crée le dossier .ssh

```
root@debiansio:~# mkdir -p /home/.ssh
```

Sur le client :

On se connecte avec chaque utilisateur puis on se place dans le répertoire /home/user et on créer le dossier .ssh

```
user1@debiansio:~$ cd /home/user1/
user1@debiansio:~$ mkdir /home/user1/.ssh
```

Gérer les clés : permission dossier

On modifie maintenant les permissions de ces fichiers pour permettre l'écriture par les bons utilisateurs dans ces dossiers.

On modifie maintenant les permissions sur chacun des dossiers des users :

```
user1@debiansio:~$ chmod 0770 ~/.ssh
```

```
user2@debiansio:~$ chmod 0770 ~/.ssh
```

```
user3@debiansio:~$ chmod 0770 ~/.ssh
```

La commande **chmod 0770** donne les permissions de lecture, écriture et exécution aux groupes et au propriétaire mais aucune permission aux autres utilisateurs.

```
root@debiansio:~# chmod 0770 /home/.ssh
```

On modifie également les permissions pour le dossier root.

Génération des clés :

On peut maintenant passer à l'étape de la génération des clés.

On choisit d'utiliser comme mot de passe sio pour générer les clés.

On génère une clé avec la commande : **ssh-keygen -t dsa -f ~/.ssh/id_dsa**

On entre ensuite un mot de passe à partir duquel sera généré la clé DSA

```
user3@debiansio:~$ ssh-keygen -t dsa -f ~/.ssh/id_dsa
Generating public/private dsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user3/.ssh/id_dsa
Your public key has been saved in /home/user3/.ssh/id_dsa.pub
The key fingerprint is:
SHA256:X66CEL3N7SwrA5L/wwBONGQ1Xn2lrFt85CQUx2+nguQ user3@debiansio
The key's randomart image is:
+---[DSA 1024]----+
| .+.o.  o+o      |
| ooo  .o.o.      |
| +++ . .+ o.     |
|  o . .o.=  o .  |
| o o ..*Soo..o   |
|  + + .oEooo.    |
|   o =.. o...    |
|    . * o o.     |
|     ..+.+.      |
+----[SHA256]-----+
user3@debiansio:~$
```

Génération des clés : Vérification :

On vérifie que les clés sont générées.

On se place dans le fichier /home/user1/.ssh. On remarque que deux clés sont générées:

```
user3@debiansio:~$ cd /home/user3/.ssh/
user3@debiansio:~/.ssh$ ls
id_dsa  id_dsa.pub
```

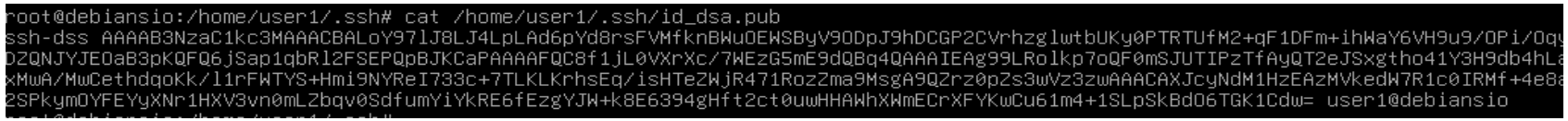
La première clé **id_dsa** est la clé privée. On peut afficher son contenue avec la commande : **cat /home/user1/.ssh/id_dsa**

```
root@debiansio:/home/user1/.ssh# cat /home/user1/.ssh/id_dsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBeqLqERd
+C72H+A+4tnneIAAAAEAAAAAEAAAGyAAAAB3NzaC1kc3MAAACBALoY97lJ8LJ4LpLAd6pY
d8rsFVMfknBWuOEWSByV9ODpJ9hDCGP2CVrhzglwtbUKy0PTRTUfM2+qF1DFm+ihWaY6VH
9u9/OPi/OqyGPFPOvP74OYMy7PuSdrBQRbMG/STX65DZQNJYJEOaB3pKQFQ6jSap1qbRl2
FSEPQpBJKCaPAAAAFQC8f1jL0VXrXc/7WEzG5mE9dQBq4QAAAIEAg99LRolkp7oQF0mSJU
TIPzTfAyQT2eJSxgtho41Y3H9db4hLakhyHLSB/ZDhHLxfHltFPoVKlyznSmv+xMwA/MwC
ethdqoKk/l1rFWTYS+Hmi9NYReI733c+7TLKLKrhsEq/isHTeZWjR471RozZma9MsgA9QZ
rz0pZs3wVz3zwAAACAXJcyNdM1HzEAzMVkedW7R1c0IRMf+4e8ajQnT6+s4I+SV9x84MLV
nDAtKevq0S222SPkymOYFEYyXNr1HXV3vn0mLZbqv0SdfumYiYkRE6fEzgYJW+k8E6394g
Hft2ct0uwHHAWhXWmECrXFYKwCu61m4+1SLpSkBdO6TGK1CdwAAAHwV1/GuG9oSGL78iMH
fW1iYdUBtPIxM1Fe6VRkX1QThI+rSu4JNU2I2Y5ioc/izYgtSzRf0iG/Ztf5J4YTlMie2b
ihBgC1cFmpjJ6SZ2VvRsrnHMbnfa0TVQaz2o2TPR4UxBkkYISVR278oepXlAqLihDyz//M
CaSQGvVmh6fGSP3JD1R8lyGdu32P2hetar3PVY2FqqZYJ4LHjI9KUs7eWKafmGZWUfgi2e
o2rf3tU6zt9l49jvIi8ciAq8vIAKtXB7fMUMDIREs+K+raviWfCeR64BnRTBo31ta4FjXU
ViTDvNXkVb9EFH+LQ5BtM3TuVR18iHZ6mvO82cOoDfp4leSP9g/yfGlfvhUOIuIO6h3Jt/
N98qlxbXFNKMFYwPBWPrKkZ6oSeOcRZc5pCWZhDuXaEt4UPUsgtuGCjRY3kh91cXoaqFwX
54dtXbrSRHDgv+tQqu4QCjXE4MO0Uvv/uNSsuNwVwrzF9Jg4/zjD9uEPY72rTHyN+LzhJQ
M427nVxdYazu47THrP/0X1Il4hLQgnMFEuCj+t/FBHy9unP2A9c8kLrzx3P7V1P2WR+k8Q
GVsaYZT9rd3crAIuCh5IS5Sb/vRFpyVtoe1AM/3KmnqcsyNUM0KXm2lORQf2R8nB5N6LhN
ubpgNTYQsU1UV5cg==
-----END OPENSSH PRIVATE KEY-----
```

Génération des clés : Vérification :

Voyons les différences entre les deux clés.

La deuxième clé, id_dsa.pub est la clé publique. On l'affiche avec la commande : **cat /home/user1/.ssh/id_dsa.pub**



La clé privée est utilisée pour prouver notre identité au serveur. On l'utilise pour déchiffrer les données chiffrées avec la clé public.

La clé publique doit être partagé au serveur. Le serveur utilise cette clé pour chiffrer les données. De plus elle est utilisée pour vérifier l'identité de l'utilisateur pour la connexion au serveur.

Envoie des Clés :

Après avoir générés les clés pour chaque user, on peut envoyer la clé publique au serveur pour qu'il puisse nous identifier.

On envoie la clé publique au serveur avec la commande : **ssh-copy-id –i ~/.ssh/id_dsa.pub** *loginuser*@*adresseIPduserveur*

```
user2@debiansio:~$ ssh-copy-id -i ~/.ssh/id_dsa.pub user2@192.168.100.75
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/user2/.ssh/
id_dsa.pub"
The authenticity of host '192.168.100.75 (192.168.100.75)' can't be established.
ED25519 key fingerprint is SHA256:q/xvkTqvnQSlDNUnA+hJ17XfdpEBLSmiGHSRUmzPeHc.
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 already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
user2@192.168.100.75's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'user2@192.168.100.75'"
and check to make sure that only the key(s) you wanted were added.
```

On accepte l'empreinte qui est la clé publique du serveur et qui sera déposé dans le fichier des hôtes connus.

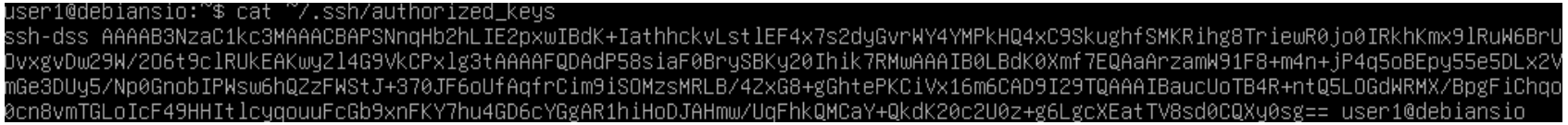
Clés autorisées :

On retrouve dans le fichier autorized_keys toutes les clés publiques associé à un utilisateur autorisé à se connecter.

On accepte l'empreinte qui est la clé publique du serveur et qui sera déposé dans le fichier des hôtes connus.

Depuis le dossier /root/home/.ssh/ on affiche les clés publiques enregistrées avec la commande : **cat authorized_keys**

```
user1@debiansio:~$ cat ~/.ssh/authorized_keys
```



Test de la connexion :

Maintenant on peut tester la connexion au serveur SSH.

Depuis un utilisateur du client dont on a envoyé la clé au serveur on essaie de se connecter.

Pour se connecter on utilise depuis l'user du client la commande :

```
user1@debiansio:~$ ssh 192.168.100.75
user1@192.168.100.75's password:
Linux debiansio 6.1.0-30-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.124-1 (2025-01-12) 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
dividual files in /usr/share/doc/*/copyright.

bian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar  4 15:14:21 2025 from 192.168.100.88
user1@debiansio:~$ exit
déconnexion
Connection to 192.168.100.75 closed.
user1@debiansio:~$
```

Avec un autre utilisateur dont on n'a pas envoyé la clé, la connexion échoue.

```
oot@debiansio:~# ssh 192.168.100.75
The authenticity of host '192.168.100.75 (192.168.100.75)' can't be established.
25519 key fingerprint is SHA256:q/xvkTqvnQSlDNUnA+hJ17XfdpEBLSmiGHSRUmzPeHc.
is key is not known by any other names.
e you sure you want to continue connecting (yes/no/[fingerprint])? yes
arning: Permanently added '192.168.100.75' (ED25519) to the list of known hosts.
oot@192.168.100.75's password:
ermission denied, please try again.
```

Autorisation avancée : groupes :

On veut autoriser uniquement les utilisateurs des groupes root et ssh à se connecter.

Pour limiter les groupes on modifie le fichier sshd_config et on ajoute la ligne AllowGroups suivi des groupes que l'on veut autoriser à se connecter. Pour nous, le groupe ssh et le groupe root.

```
root@debiansio:~# nano /etc/ssh/sshd_config
```

```
AllowGroups ssh root
```

On relance le serveur pour appliquer les changements

```
root@debiansio:~# service ssh stop
root@debiansio:~# service ssh start
```

Autorisation avancée : groupes :

On vérifie en essayant de se connecter avec l'user1 membre de ssh et l'user3 qui n'en n'est pas membre.

Les autorisations pour le groupe sont bien prises en compte.

L'user3 se voit refuser la connexion

```
user3@debiansio:~$ ssh 192.168.100.94
user3@192.168.100.94's password:
Permission denied, please try again.
```

L'user1 peut par contre toujours se connecter

```
user1@debiansio:~$ ssh 192.168.100.94
user1@192.168.100.94's password:
Linux debiansio 6.1.0-30-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.124-1 (2025-01-12) 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: Wed Mar  5 11:21:14 2025 from 192.168.60.34
user1@debiansio:~$ exit
déconnexion
Connection to 192.168.100.94 closed.
```

Autorisation avancée : clés SSH :

On veut également faire en sorte que les utilisateurs ne se connectent que avec une clé SSH.

Pour gérer la méthode d'authentification, on ajoute dans le fichier sshd_config la ligne **PasswordAuthentification no**

```
root@debiansio:~# nano /etc/ssh/sshd_config
```

```
PasswordAuthentication no
```

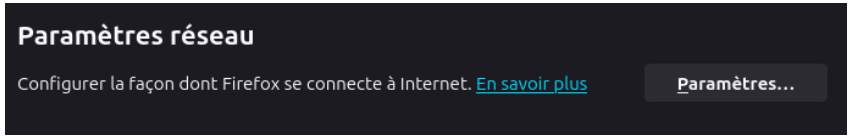
Un utilisateur sans clé se voit bien refusé l'accès.

```
root@debiansio:~# ssh 192.168.100.94
root@192.168.100.94: Permission denied (publickey).
```

Les plus du SSH :

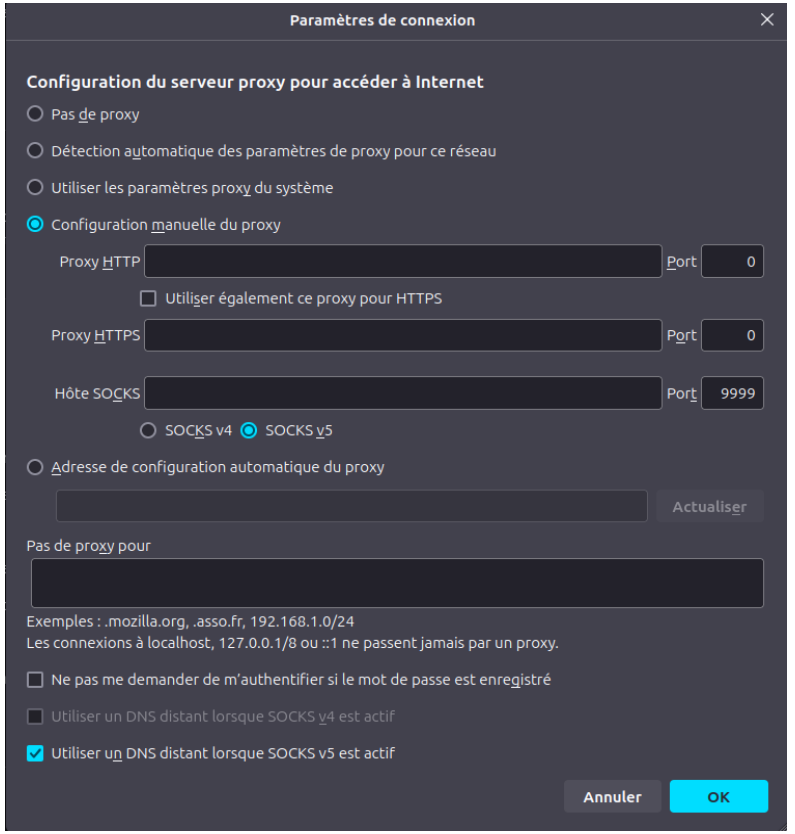
On établit un tunnel proxy entre notre machine local et le serveur SSH.

Configuration du client Web :



Dans les paramètres réseaux du navigateur firefox, on passe en configuration manuelle du proxy et sur la section hôte SOCKS on entre le port 9999.

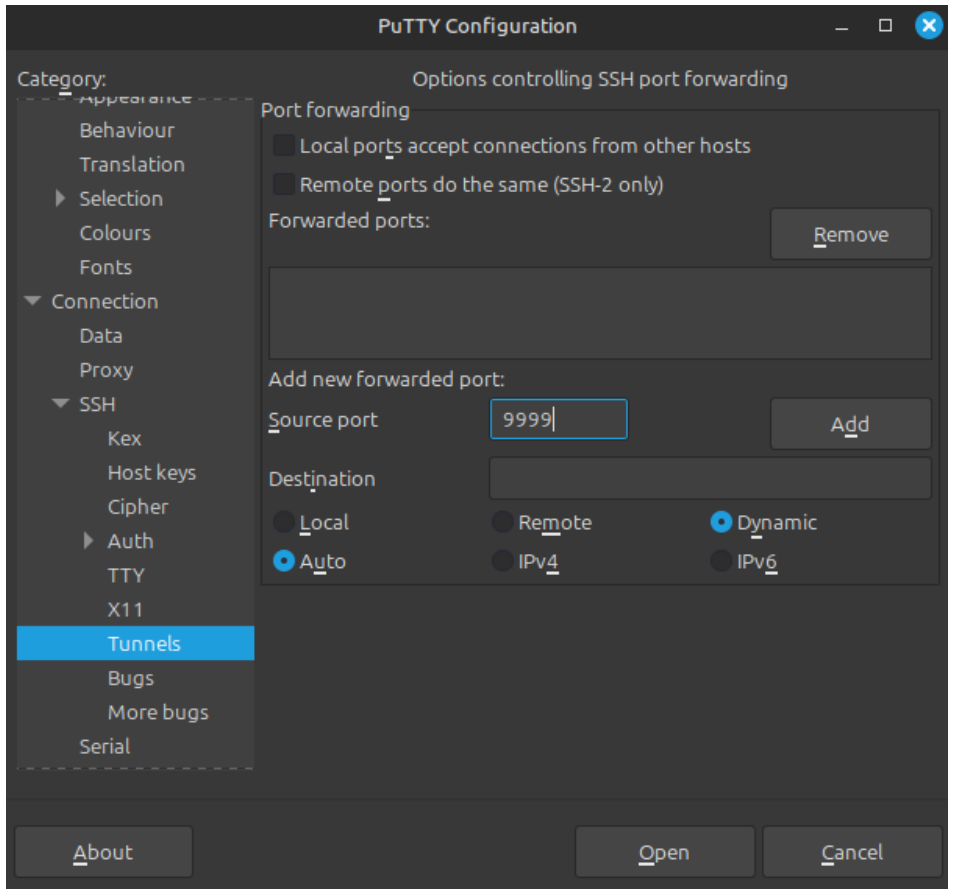
On coche également la case utiliser un DNS lorsque SOCKS 5 actif.



Les plus du SSH :

On configure maintenant Putty.

Configuration du client Web :



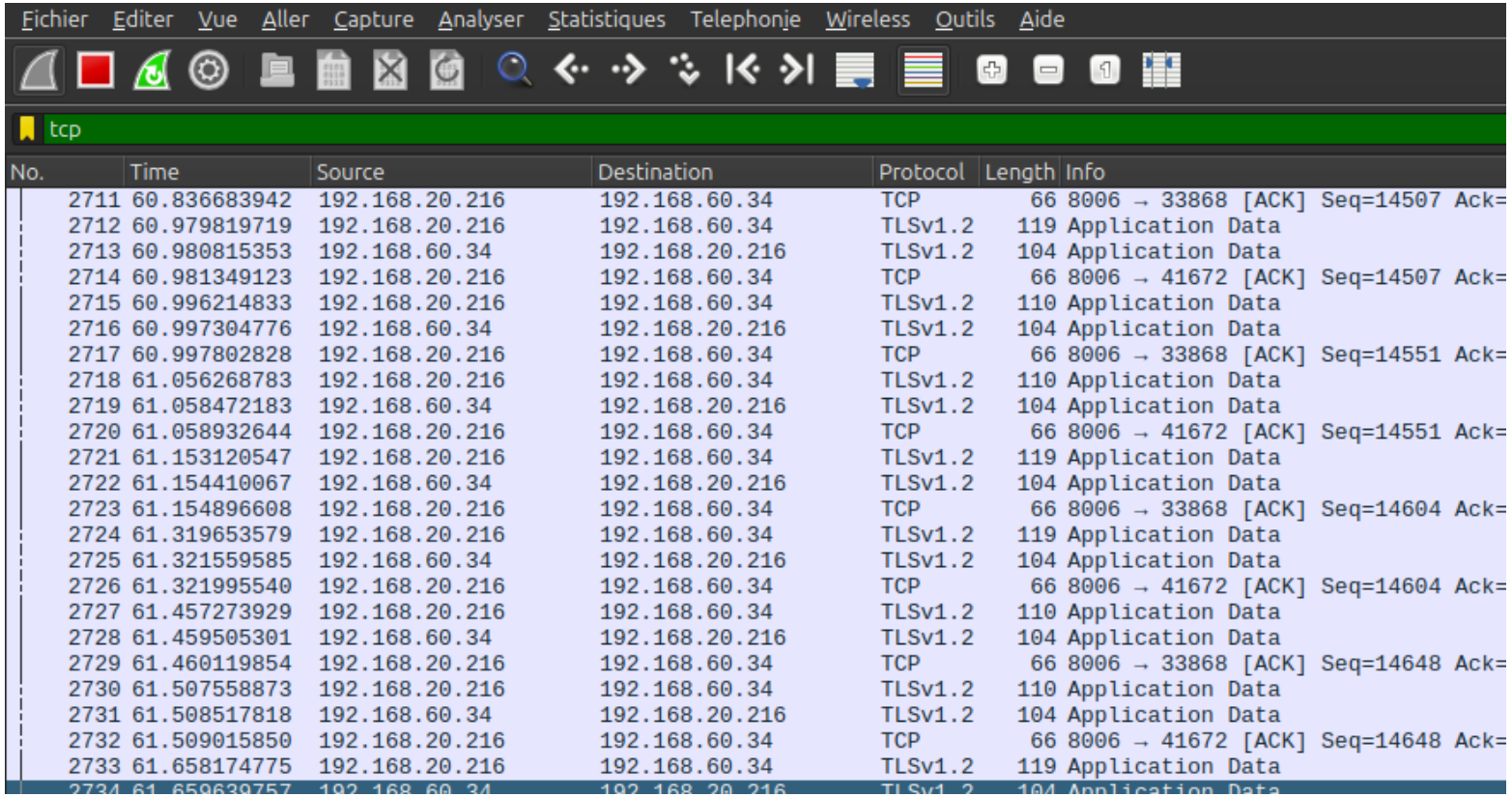
Sous la catégorie SSH, on va dans Tunnels et on définit la destination en auto et en Dynamic puis on entre dans source port 9999.

On peut ensuite se connecter au serveur avec putty comme au début duTP.

Les plus du SSH :

On capture la trame réseau avec wireshark.

Pour plus d'informations sur wireshark, voir TP B1 : Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
2711	60.836683942	192.168.20.216	192.168.60.34	TCP	66	8006 → 33868 [ACK] Seq=14507 Ack=
2712	60.979819719	192.168.20.216	192.168.60.34	TLSv1.2	119	Application Data
2713	60.980815353	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data
2714	60.981349123	192.168.20.216	192.168.60.34	TCP	66	8006 → 41672 [ACK] Seq=14507 Ack=
2715	60.996214833	192.168.20.216	192.168.60.34	TLSv1.2	110	Application Data
2716	60.997304776	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data
2717	60.997802828	192.168.20.216	192.168.60.34	TCP	66	8006 → 33868 [ACK] Seq=14551 Ack=
2718	61.056268783	192.168.20.216	192.168.60.34	TLSv1.2	110	Application Data
2719	61.058472183	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data
2720	61.058932644	192.168.20.216	192.168.60.34	TCP	66	8006 → 41672 [ACK] Seq=14551 Ack=
2721	61.153120547	192.168.20.216	192.168.60.34	TLSv1.2	119	Application Data
2722	61.154410067	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data
2723	61.154896608	192.168.20.216	192.168.60.34	TCP	66	8006 → 33868 [ACK] Seq=14604 Ack=
2724	61.319653579	192.168.20.216	192.168.60.34	TLSv1.2	119	Application Data
2725	61.321559585	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data
2726	61.321995540	192.168.20.216	192.168.60.34	TCP	66	8006 → 41672 [ACK] Seq=14604 Ack=
2727	61.457273929	192.168.20.216	192.168.60.34	TLSv1.2	110	Application Data
2728	61.459505301	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data
2729	61.460119854	192.168.20.216	192.168.60.34	TCP	66	8006 → 33868 [ACK] Seq=14648 Ack=
2730	61.507558873	192.168.20.216	192.168.60.34	TLSv1.2	110	Application Data
2731	61.508517818	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data
2732	61.509015850	192.168.20.216	192.168.60.34	TCP	66	8006 → 41672 [ACK] Seq=14648 Ack=
2733	61.658174775	192.168.20.216	192.168.60.34	TLSv1.2	119	Application Data
2734	61.659639757	192.168.60.34	192.168.20.216	TLSv1.2	104	Application Data

Après avoir filtré les protocoles en TCP, on peut observer que la VM du server SSH et la machine cliente communiquent bien entre elles.

Les plus du SSH :

On fait une analyse du réseau avec nmap.

Pour plus d'informations voir TP B3 : Identifier les menaces

Avec un nmap on observe que les ports 22 et 80 sont ouverts et ainsi que le serveur ssh et le serveur web sont bien à l'écoute.

```
maxence@maxence-ThinkPad-P16s-Gen-2:~$ nmap -v -A 192.168.100.94
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-05 13:23 CET
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 13:23
Completed NSE at 13:23, 0.00s elapsed
Initiating NSE at 13:23
Completed NSE at 13:23, 0.00s elapsed
Initiating NSE at 13:23
Completed NSE at 13:23, 0.00s elapsed
Initiating Ping Scan at 13:23
Scanning 192.168.100.94 [2 ports]
Completed Ping Scan at 13:23, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:23
Completed Parallel DNS resolution of 1 host. at 13:23, 0.01s elapsed
Initiating Connect Scan at 13:23
Scanning 192.168.100.94 [1000 ports]
Discovered open port 80/tcp on 192.168.100.94
Discovered open port 22/tcp on 192.168.100.94
```

L'adresse IP 192.168.100.94 est celle du serveur SSH