SERVICE SSH

Linux CentOS

Table des matières

1 Introduction

De nombreux outils ont été fournis pour utiliser la capacité du réseau. Échanger, copier, utiliser des Shells à distance. Les noms de ces outils sont respectivement **ftp**, **rcp**, **telnet**, etc... Bien que ces outils, utilisés pendant des années et même encore aujourd'hui dans beaucoup d'entreprises, soient très pratiques, ils comportent une faiblesse importante. Leurs transactions sont transmises en clair via le réseau. De ce fait, n'importe quelle personne mal intentionnée peut être en mesure d'observer ce que vous faites, allant même jusqu'à subtiliser vos données personnelles et mots de passe.

SSH signifie **Secure SHell**. C'est un protocole qui permet de faire des connexions sécurisées (cryptées) entre un serveur et un client **SSH**. Nous allons utiliser le programme OpenSSH, qui est la version libre du client et du serveur **SSH**. Le service SSH écoute sur le port 22.

2 Installation

Pour **CentOS**, les paquetages **openssh** sont les suivants :

- **openssh**
- **openssh-askpass**
- **openssh-clients (client ssh)**
- **openssh-server (serveur ssh)**

Le client et le serveur SSH sont installés par défaut :

```
[root@localhost ~]# rpm -q openssh-clients
openssh-clients-7.4p1-16.el7.x86_64
```

```
[root@localhost ~]# rpm -q openssh-server
openssh-server-7.4p1-16.el7.x86_64
```

3 Test de connexion

À partir du serveur SSH :

```
 [root@ServeurSSH ~]# ssh localhost

The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is
SHA256:vKZgcVIHOjXNV/u7zrH4WDhtJpWRp6MbVNBWZW7phuI.
ECDSA key fingerprint is
MD5:22:84:27:90:73:6e:21:84:0b:ef:ab:88:45:39:88:a2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known
hosts.
root@ServeurSSH's password: *******
Last login: Wed Apr 24 21:04:02 2019 from 192.168.17.1
[root@ServeurSSH ~]#
```

Pour fermer la session SSH :

```
[root@ServeurSSH ~]# exit
[root@ServeurSSH ~]#
```

À partir du réseau (client SSH) :

```
[root@ClientSSH ~]# ssh 192.168.17.131

The authenticity of host '192.168.17.131 (192.168.17.131)' can't be
established.
ECDSA key fingerprint is
SHA256:vKZgcVIHOjXNV/u7zrH4WDhtJpWRp6MbVNBWZW7phuI.
ECDSA key fingerprint is
MD5:22:84:27:90:73:6e:21:84:0b:ef:ab:88:45:39:88:a2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.17.131' (ECDSA) to the list of
known hosts.
root@192.168.17.131's password: *******
Last login: Wed Apr 24 21:16:32 2019 from localhost
[root@ServeurSSH ~]#
```

Afficher les sessions établies :

```
[root@ServeurSSH ~]# netstat -taupen|grep EST

tcp 0 0 192.168.17.131:22 192.168.17.1:51460    ESTABLISHED 0 41173  7335/sshd: root@pts
tcp 0 0 192.168.17.131:22 192.168.17.130:44994  ESTABLISHED 0 41839  7409/sshd: root@pts
[root@ServeurSSH ~]#
```

4 Configuration du serveur SSH

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**.

Les lignes les plus importantes de ce fichier de configuration sont:

```
Port 22
```

Signifie que le serveur **SSH** écoute sur le port **22**, qui est le port par défaut de **SSH**. Vous pouvez le faire écouter sur un autre port en changeant cette ligne. Vous pouvez aussi le faire écouter sur plusieurs ports à la fois en rajoutant des lignes similaires.

```
Protocol 2
```

Signifie que votre serveur **SSH** accepte uniquement la version 2 du protocole **SSH**. C'est une version plus sécurisée que la version 1 du protocole. Certains vieux clients **SSH** utilisent **SSH** version 1. Si vous voulez que le serveur accepte les deux protocoles, changez la ligne en :

```
Protocol 2,1
```

```
PermitRootLogin yes
```

Signifie que vous pouvez ouvrir une session **SSH** en tant que **root**.

Vous pouvez mettre "***no***", ce qui signifie que pour vous ne pouvez pas ouvrir une session **SSH** en tant que **root**. Il faut d'abord ouvrir une session avec un utilisateur standard, ensuite utiliser la commande **su** pour devenir **root**.

```
X11Forwarding yes
```

Signifie que vous allez pouvoir travailler en export display par **SSH**. Ce sera expliqué plus tard, dans la partie **Xforwarding**.

Si vous avez modifié le fichier de configuration du serveur, il faut redémarrer le service **SSH**.

5 Contrôle du service SSH

Démarrer le service SSH

```
[root@localhost ~]#  systemctl start sshd
```

Redémarrer le service SSH

```
[root@localhost ~]#  systemctl restart sshd
```

Vérifier le statut du service SSH

```
[root@localhost ~]#  systemctl status sshd
```

Arrêter le service SSH

```
[root@localhost ~]#  systemctl stop sshd
```

6 Journalisation

Si vous rencontrer un problème, vous pouvez consulter le fichier log :

/var/log/secure

```
[root@localhost ~]#  tail  -f  /var/log/secure
```

7 Client SSH

Le client **SSH** est l'outil (commande *ssh*) qui nous permet de se connecter au serveur **SSH**. Sa syntaxe est la suivante :

```
ssh -vv -l utilisateur -p port -(1|2) hôte  ou    ssh utilisateur@hôte
```

Voici la description des options :

Option	Description
-l login	Identifiant de l'utilisateur.
-v -vv -vvv	Mode verbeux, permet d'obtenir les messages de débogage plus ou moins complets (le nombre maximum étant 3).
-1 ou -2	Version de **ssh :** ssh1 ou ssh2
-p port	Numéro du port distant

Le fichier de configuration du client **SSH** est :

```
/etc/ssh/ssh_config
```

8 Authentification

Il est possible de s'authentification de trois façons **SSH**.

8.1 Authentification par mot de passe

C'est la méthode standard pour ouvrir une session SSH :

```
[root@ClientSSH ~]# ssh 192.168.17.131

root@192.168.17.131's password: *******
Last login: Wed Apr 24 21:17:49 2019 from 192.168.17.130
[root@ServeurSSH ~]#
```

On peut spécifier le nom de l'usager avec l'option **-l** :

```
[root@ClientSSH ~]# ssh -l hakimb 192.168.17.131

hakimb@192.168.17.131's password: *******
[hakimb@ServeurSSH ~]$
```

Il est aussi possible d'utiliser le @ :

```
[root@ClientSSH ~]# ssh hakimb@192.168.17.131

hakimb@192.168.17.131's password: *******
[hakimb@ServeurSSH ~]$
```

Si c'est la première connexion **SSH** depuis ce client, il vous demande si le **fingerprint** de la clé publique présentée par le serveur est bien le bon.

Pour être sûr que vous vous connectez au bon serveur, vous devez connaître de façon certaine le **fingerprint** de sa clé publique et la comparer à celle qu'il vous affiche.

Si les deux **fingerprints** sont identiques, répondez ***yes***, et la clé publique du serveur est alors rajoutée au fichier **~/.ssh/known_hosts**

Si vous vous êtes déjà connecté depuis ce client vers le serveur, sa clé publique est déjà dans le fichier **~/.ssh/known_hosts** et il ne vous demande donc rien.

8.2 Authentification par clé (avec passphrase)

Au lieu de s'authentifier par mot de passe, les utilisateurs peuvent s'authentifier grâce à la cryptographie asymétrique et son couple de clés privée/publique, comme le fait le serveur **SSH** auprès du client **SSH**.

1) Générer la paire de clés (sur le client SSH)

L'utilisateur doit avoir une clé publique et une clé privée. On doit transférer la clé publique de l'utilisateur sur le serveur SSH.

Pour générer une paire de clés **DSA** de l'utilisateur **root** :

```
[root@ClientSSH ~]# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa):
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
SHA256:fbNRxhlOXwHzBapJnV7Tx0ZMoZRfu/zmaEgeWS1AxjU
root@ClientSSH
The key's randomart image is:
+---[DSA 1024]----+
|           oo=E*B|
|           o+B+O=|
|          . +oXoO|
|         o + +o=o|
|        S + =o...|
|           .++ o |
|           o.o  .|
|            o ..o|
|              ..o.|
+----[SHA256]-----+
[root@ClientSSH ~]#
```

Pour générer une paire de clés **RSA** de l'utilisateur **root** :

```
 [root@ClientSSH ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:iSrb1pts6uVMjBCme36ZTxSrIeR+/GruRAS5YpBTRIw
root@ClientSSH
The key's randomart image is:
+---[RSA 2048]----+
| B=.             |
|E o.             |
|..+.. .          |
|.*.o   + .       |
|o.+ o + S        |
| o = B           |
|. + B+=          |
| o **Oo.         |
|  oBB=O.         |
+----[SHA256]-----+
```

Pour les deux algorithmes (**DSA**, **RSA**), le système nous demande dans quel fichier nous désirons sauvegarder la clé. Par défaut la clé privée est stockée dans le fichier **~/.ssh/id_dsa** avec les permissions **600** et la clé publique est stockée dans le fichier **~/.ssh/id_dsa.pub** avec les permissions **644**.

```
[root@ClientSSH ~]# ls -l .ssh/
total 12
-rw------- 1 root root 1766 Apr 25 05:24 id_rsa
-rw-r--r-- 1 root root  396 Apr 25 05:24 id_rsa.pub
-rw-r--r-- 1 root root  176 Apr 24 06:08 known_hosts
```

Par la suite, une **passphrase** nous est demandée. Celle-ci est un « mot de passe amélioré », car non limité à un mot ou une petite suite de caractères. Il faut cependant prendre des précautions, car en cas de perte de la **passphrase**, vous ne pourriez plus vous authentifier en tant que propriétaire authentique.

La **passphrase** permet de protéger la clé privée. Elle sera demandée à chaque utilisation de la clé privée. Un mécanisme appelé **ssh-agent** permet de ne pas rentrer le **passphrase** à chaque connexion.

Vous pouvez à tout moment changer la **passphrase** qui protège votre clé privée avec la commande :

```
[root@ClientSSH ~]# ssh-keygen -t dsa -p
Enter file in which the key is (/root/.ssh/id_dsa):
Enter old passphrase: *******
Enter new passphrase (empty for no passphrase): *******
Enter same passphrase again: *******
Your identification has been saved with the new passphrase.
[root@ClientSSH ~]#
```

2) Autoriser votre clé publique (sur le serveur SSH)

Pour cela, il suffit de copier votre clé publique dans le fichier **~/.ssh/authorized_keys** du serveur SSH.

La commande suivante permet de réaliser cette opération via **SSH** :

```
[root@ClientSSH ~]# ssh-copy-id root@192.168.17.131

/usr/bin/ssh-copy-id: INFO: Source of key(s) to be
installed: "/root/.ssh/id_rsa.pub"
/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
root@192.168.17.131's password: *******

Number of key(s) added: 1

Now try logging into the machine, with:
"ssh 'root@192.168.17.131'"
and check to make sure that only the key(s) you wanted were
added.

[root@ClientSSH ~]#
```

On peut maintenant se connecter au serveur SSH sans le mot de passe de l'utilisateur. Il faut juste fournir la **passphrase** :

```
[root@ClientSSH ~]# ssh root@192.168.17.131

Enter passphrase for key '/root/.ssh/id_rsa':*******
Last login: Wed Apr 24 21:36:17 2019 from 192.168.17.130
[root@ServeurSSH ~]#
```

8.3 Authentification par clé (avec passphrase + agent)

Pour éviter d'avoir à saisir la **passphrase** à chaque ouverture de session SSH, il suffit de fournir la **passphrase** à l'agent **ssh-agent** qui va à son tour la fournir au client SSH au besoin. La commande **ssh-add** permet de fournir la **passphrase** à **ssh-agent**.

Pour démarrer l'agent ssh

```
 [root@ClientSSH ~]# ssh-agent

SSH_AUTH_SOCK=/tmp/ssh-apXJvlHqqNlR/agent.7177; export SSH_AUTH_SOCK;
SSH_AGENT_PID=7178; export SSH_AGENT_PID;
echo Agent pid 7178;
```

Exécuter les trois dernières lignes (faire un copier/coller):

```
[root@ClientSSH ~]# SSH_AUTH_SOCK=/tmp/ssh-
apXJvlHqqNlR/agent.7177; export SSH_AUTH_SOCK;

[root@ClientSSH ~]# SSH_AGENT_PID=7178; export SSH_AGENT_PID;

[root@ClientSSH ~]# echo Agent pid 7178;
Agent pid 7178
```

Pour transmettre la **passphrase** à l'agent ssh:

```
[root@ClientSSH ~]# ssh-add
Enter passphrase for /root/.ssh/id_rsa: *******
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
Enter passphrase for /root/.ssh/id_dsa: *******
Identity added: /root/.ssh/id_dsa (/root/.ssh/id_dsa)
[root@ClientSSH ~]#
```

Il vous demande alors votre **passphrase**. Maintenant que votre clé a été transmise à l'agent, vous pouvez vous connecter sans entrer de mot de passe à toutes les machines pour lesquelles vous avez transmis votre clé publique dans le fichier **~/.ssh/authorized_keys**.

```
[root@ClientSSH ~]# ssh root@192.168.17.131
Last login: Wed Apr 24 22:10:08 2019 from 192.168.17.130
[root@ServeurSSH ~]#
```

Pour arrêter l'agent **ssh-agent**:

```
[root@ClientSSH ~]# ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 7178 killed;
```

Exécuter les trois dernières lignes (faire un copier/coller):

```
[root@ClientSSH ~]# unset SSH_AUTH_SOCK;
[root@ClientSSH ~]# unset SSH_AGENT_PID;
[root@ClientSSH ~]# echo Agent pid 7178 killed;
Agent pid 7178 killed
[root@ClientSSH ~]#
```

9 La copie sécurisée

SSH fournit un outil de copie sécurisée en standard, sous le nom de **scp** pour **SecureCoPy**. Il remplace son ancêtre **rcp**. Son usage est très simple :

scp hôte_source:source hôte_destination:destination

Lorsque l'hôte correspond à la machine où vous vous trouvez, il n'est pas nécessaire de l'inscrire.

```
[root@clientssh ~]# touch fichier.doc

[root@clientssh ~]# scp fichier.doc 192.168.17.131:/root
Enter passphrase for key '/root/.ssh/id_rsa': : *******
fichier.doc                   100%    0     0.0KB/s   00:00
```

Vous pouvez également faire des copies récursives, comme nous le ferions avec n'importe quel autre utilitaire de copie. Par exemple pour copier le répertoire **/data** dans **/root/data** du serveur **192.168.17.131** :

```
[root@clientssh ~]# mkdir /data
[root@clientssh ~]# touch /data/f1 /data/f2 /data/f3/

[root@clientssh ~]# scp -r /data/ 192.168.17.131:/root
Enter passphrase for key '/root/.ssh/id_rsa': *******
f1                            100%    0     0.0KB/s   00:00
f2                            100%    0     0.0KB/s   00:00
f3                            100%    0     0.0KB/s   00:00
[root@clientssh ~]#
```

10 Le transfert de fichier sécurisé

Tout comme on peut copier des fichiers à distance par l'intermédiaire de **scp**, il est également possible de transférer des fichiers par l'intermédiaire d'un ftp sécurisé nommé **SecureFTP**.

```
[root@clientssh ~]# sftp hakimb@192.168.17.131

hakimb@192.168.17.131's password: *******
Connected to 192.168.17.131.
sftp>
```

```
sftp> lpwd
Local working directory: /root

sftp> pwd
Remote working directory: /home/hakimb

sftp> put anaconda-ks.cfg
Uploading anaconda-ks.cfg to /home/hakimb/anaconda-ks.cfg
anaconda-ks.cfg                100% 1260    47.9KB/s   00:00

sftp> bye
[root@clientssh ~]#
```

Commandes disponibles sur **sftp** :

Commande	Description
cd path	Change le répertoire distant vers 'path'
lcd path	Change le répertoire local vers 'path'
chgrp grp path	Change le groupe de fichier 'path' par 'grp'
chmod mode path	Change les permissions du fichier 'path' à 'mode'
chown own path	Change le propriétaire du fichier 'path' par 'own'
help	Affiche ce message d'aide
get remote-path [local-path]	Télécharge le fichier
lls [ls-options [path]]	Affiche le listing du répertoire local
ln oldpath newpath	Crée un lien symbolique du fichier distant
lmkdir path	Crée un répertoire local
lpwd	Affiche le répertoire courant
ls [path]	Affiche le listing du répertoire distant
lumask umask	Positionne l'umask local à 'umask'
mkdir path	Crée le répertoire distant
put local-path [remote-path]	Charge le fichier
pwd	Affiche le répertoire courant distant
exit	Quitte sftp
quit	Quitte sftp
rename oldpath newpath	Renomme le fichier distant
rmdir path	Supprime le répertoire distant
rm path	Supprime le fichier distant
symlink oldpath newpath	Crée un lien symbolique du fichier distant
version	Affiche la version de sftp
!command	Exécute la 'commande' dans un Shell local
!	Sort vers un Shell local
?	Affiche ce message d'aide