ssh-client : se connecter pour administrer à distance

- Objet : Utilisation de ssh comme client
- Niveau requis :
  debutant,avisé
- Commentaires : *Administrer son serveur à distance, établir un tunnel sécurisé pour relayer des ports, etc...*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  à-tester
  - Création par MaTTuX_ le 17/06/2007
  - Testé par <...> le <...>
- Commentaires sur le forum : ici[1)]

Introduction

Lorsque vous démarrez un ordinateur, à moins que vous n'ayez configuré de connexion automatique, il vous est demandé un nom d'utilisateur et un mot de passe.
C'est le processus de *login* qui vous permet de vous authentifier en tant qu'utilisateur de la machine.
À l'issue de cette authentification, vous avez accès à une interface vous permettant de lancer des commandes, souvent une interface graphique comme shell.

Le principe ici va être le même :
via un canal sécurisé[2)] vous allez fournir

1. un nom d'utilisateur et
2. un mot de passe

mais cette fois, depuis votre PC à une machine distante.

Installation sur le poste client

Comme d'habitude, ça tient en une ligne :

```
apt-get update && apt-get install openssh-client
```

Utilisation

Pour pouvoir accéder à une machine via la couche IP (qu'utilise SSH), il nous faut :

- soit l'adresse IP de la machine,
- soit son nom d'hôte (hostname).

Tips pour connaitre l'adresse IP d'une machine :

```
hostname -I
```

Puis pour son nom d'hôte (hostname) :

```
hostname
```

Par exemple, pour se connecter au serveur ssh **debian**, vous pouvez lui envoyer des ping en l'adressant par son IP ainsi :

```
ping -c 5 128.31.0.51
```

Des ping depuis un autre ordi du réseau local ? 128.31.0.51 est une adresse machine, non ? Peut être faut il préciser qu'il faut l'adresse publique ?? ou via son nom d'hôte :

```
ping -c 5 debian.org
```

Obtenir un shell distant

Par exemple, pour vous connecter sous le nom d'utilisateur[3] *jojo* sur la machine dont le nom (hostname) est *coincoin*[4], vous faites simplement :

```
ssh jojo@coincoin
```

Forcer le client SSH à utiliser IPv4 ou IPv6

```
 ssh -6 jojo@coincoin
```

où alors

```
 ssh -4 jojo@coincoin
```

encore une fois comment est ce possible sans adresse publique ?? Si la machine sur laquelle le serveur serveur ssh tourne n'écoute pas sur le port 22, vous pouvez spécifier le numéro de port à l'aide de l'option `-p` :

```
ssh -p 1234 jojo@coincoin
```

Obtenir un shell distant avec export X

Il est aussi possible de lancer sur le serveur distant une application graphique qui s'affichera sur votre ordinateur client.

Pour vous connecter sous le nom d'utilisateur *jojo* sur la machine dont le nom (hostname) est *coincoin* et obtenir l'export X,

1. vous faites simplement :

```
ssh -X jojo@coincoin
```

2. Puis vous lancez votre application graphique via le shell obtenu.

Authentification

Par nom d'utilisateur/mot de passe

Le nom d'utilisateur est spécifié dans la commande de connexion. Le serveur nous demande alors notre mot de passe pour nous connecter, de la même façon que sur une machine locale.

```
ssh jojo@coincoin
```

Ce à quoi le serveur vous répond :

```
  jojo@coincoin's password:
```

Par clés asymétriques

Les mots de passe sont souvent peu complexes. On peut alors utiliser une paire de clés publique/privée plus complexe et plus sûre.

Le principe est le suivant, on génère une paire de clés. La clé privée reste sur votre poste client. La clé publique doit être transférée sur le serveur manuellement ou par le réseau. C'est cette clé qui servira alors à l'authentification. **c'est qui “on ” : le client ou le serveur ??**

La contine du captnfab :
La *clé publique* c'est un peu la serrure.
La *clé privée* est la clé.
Tu peux distribuer ta serrure à tout le monde et quand tu viens, tu viens avec ta clé et ça ouvre les portes.

Création de la paire de clés

D'aprés l'ANSSI les clefs dsa sont à éviter (cf page 7 du document). Il est recommandé d'utiliser des clefs ecdsa en 256 ou rsa en 2048. Les clés doivent être générées dans un contexte où la source d’aléa est fiable, ou à défaut dans un environnement où suffisamment d’entropie a été accumulée. Pour des raisons évidentes, les systèmes nouvellement démarrés ne peuvent avoir accumulé suffisamment d’entropie. Il est donc recommandé d’obtenir de l’aléa une fois qu’une période d’activité suffisante s’est écoulée.

A partir de la version 7 de openssh (présente dès Debian 9), le support des clés dsa est désactivé par défaut[5)].

Les commandes suivantes sont à lancer en utilisant le compte utilisateur dont vous souhaitez pouvoir ouvrir une session distante.

Pour une clef ecdsa en 256:

```
ssh-keygen -t ecdsa -b 256
```

Peut on lancer cette commande seule pour se connecter ?

Pour une clef rsa en 2048:

```
ssh-keygen -t rsa -b 2048
```

Pour une clef dsa en 2048:

```
ssh-keygen -t dsa -b 2048
```

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
/home/user/.ssh/nom_fichier
Enter passphrase (empty for no passphrase): (mettre une passphrase qui sera
demandée à chaque connexion - ou rien si ce n'est pas nécessaire pour vous)
```

Vous validez 2 fois et vous obtiendrez le résultat un rapport détaillé concernant la création de votre paire de clés dsa.

Exportation de la clé

- exportation de la clé avec la commande ssh-copy-id :

Votre clé sera générée en 1024 bits dans votre dossier /home/votre_user/.ssh/

Il a été créé deux fichiers :

```
-rw-------  1 user user 1823 mai   23 19:19 nom_fichier
-rw-r--r--  1 user user  393 mai   23 19:19 nom_fichier.pub
```

Il faut envoyer sur le serveur le fichier “nom_fichier.pub”.

Pour la mettre en place il vous suffit de la copier sur votre pc distant. **serveur ou client ?**

```
ssh-copy-id user@192.168.x.x
```

Par défaut, le client ssh s'attend à un nom de clé id_rsa.pub ou id_dsa.pub ...
L'option -i permet de forcer avec un autre nom que celui par défaut.

```
ssh-copy-id -i /home/user/.ssh/nom_fichier.pub user@192.168.x.x
```

- On vous demande alors votre mot de passe une dernière fois **on est côté serveur j'imagine ?**

Si vous avez changé côté serveur, le port 22, par exemple par le port 1234, il faut alors utiliser la commande :

```
ssh-copy-id -i ~/.ssh/id_dsa.pub -p 1234 ${USER}@192.168.x.x
```

Adaptez l'adresse IP selon celle de votre serveur.

À présent, c'est la passphrase qui sera demandée.

À savoir **la commande ssh-copy-id** qui vient d'être utilisée pour copier la clé publique, est un script qui crée à cette occasion le fichier de type répertoire **~/.ssh** et le fichier **~/.ssh/authorized_keys**.

Pour copier le contenu du fichier ~/.ssh/id_dsa.pub (créée côté client), sur le serveur par un autre moyen que la commande ssh-copy-id,
il faudra créer soi-même le fichier de type répertoire **~/ssh** et vérifier qu'il ait les droits 700 (lecture + écriture + exécution pour le propriétaire du répertoire).

- Puis soit créer le fichier **~/.ssh/authorized_keys** (`cp /chemin/de/la/clé ~/.ssh/authorized_keys`)
- Soit, s'il existe déjà parce qu'il a été créé pour un autre client qui se connecte aussi à ce serveur, y ajouter cette clé :
  `cat /chemin/de/id_rsa.pub » ~/.ssh/authorized_keys`
- enfin donner ou vérifier que “authorized_keys” ait les droits 400 (lecture seule pour le propriétaire du fichier)

- copie de la clé publique en utilisant une clef USB :

On peut par exemple copier sur une clé USB le fichier `~/.ssh/id_dsa.pub` contenant la clé publique du client, puis copier le contenu de ce fichier dans le fichier `~/.ssh/authorized_keys` sur le serveur. Pour ce faire :

- sur l'ordinateur client :

```
mount /dev/sdxx /mnt/usb
```

```
cp ~/.ssh/id_dsa.pub /mnt/usb/
```

```
umount /mnt/usb
```

-sur le serveur :

```
mount /dev/sdxx /mnt/usb
```

puis :

```
mkdir ~/.ssh
```

-puis copie de ssh/id_dsa.pub et création du fichier ~/.ssh/authorized_keys

```
mv /mnt/usb/ssh/id_dsa.pub ~/.ssh/authorized_keys
```

```
umount /mnt/usb
```

Pour vérification :

```
ls -la ~/.ssh/
```

retour de la commande

```
drwxr-xr-x  2 user user 4096 mai   15 11:05 .
drwxr-xr-x 29 user user 4096 mai   15 11:09 ..
-rw-r--r--  1 user user  402 mai   15 10:58 authorized_keys
```

- Pour donner au propriétaire du fichier uniquement les droits de lecture au fichier ~/.ssh/authorized_keys :

```
chmod 400 ~/.ssh/authorized_keys
```

retour de la commande

```
drwxr-xr-x  2 user user 4096 mai   15 11:05 .
drwxr-xr-x 29 user user 4096 mai   15 11:09 ..
-r--------  1 user user  402 mai   15 10:58 authorized_keys
```

- la connexion se fait comme précédemment :

```
ssh -p <Numéro de port choisi> user@192.168.XXXX
```

A présent, lors de votre connexion, seule votre éventuelle *passphrase* indiquée lors de la génération de la clé vous sera demandée, mais plus votre mot de passe.

Pour éviter de taper toute cette commande, il faut renseigner le fichier *~/.ssh/config* comme indiqué plus bas

ssh-agent

Afin d'éviter de retaper votre *passphrase* à chaque connexion, vous pouvez utiliser *ssh-agent*. Il doit être invoqué au début de votre session de la façon suivante avec un *shell bourne* (comme *bash*)

```
eval `ssh-agent -s`
```

et comme cela avec un *shell C*

```
eval `ssh-agent -c`
```

Il suffit ensuite d'indiquer la clé à utiliser grâce à la commande *ssh-add*

```
ssh-add $cle_privee
```

Sans l'argument *$clee_privee*, *ssh-add* ajoute les clés suivantes *~/.ssh/id_rsa, ~/.ssh/id_dsa, ~/.ssh/id_ecdsa and ~/.ssh/identity*

Pour lister les clés ajoutées

```
ssh-add -l
```

Pour supprimer toutes les clés enregistrées

```
ssh-add -D
```

Ce système ne fonctionne que pour une seule session. Ainsi, si vous fermez votre session (ou votre émulateur de terminal), votre *passphrase* vous sera à nouveau demandé. Pour garder une trace de cet enregistrement, il faut utiliser *keychain*

keychain

Installation

```
apt-get install keychain
```

configuration

Il faut appeler le script. Dans votre fichier *~/.bashrc* ou *~/.bash_profile* ajoutez (en remplaçant $cle_privee par le nom de votre clé)

```
##############################################################################
# allow $USER to use keys. Only enter once and it will remain enabled till
# you delete it or reboot the server
##############################################################################
/usr/bin/keychain $HOME/.ssh/$cle_privee
source $HOME/.keychain/$HOSTNAME-sh
```

Si votre but est uniquement de permettre à des scripts (lancés par *cron* par exemple)

de se connecter en *ssh* en interdisant une connexion manuelle sans *passphrase*, il faut utiliser le code suivant

```
##############################################################
####################
### The --clear option make sure Intruder cannot use your
existing SSH-Agents keys
### i.e. Only allow cron jobs to use password less login
##############################################################
####################
/usr/bin/keychain --clear $HOME/.ssh/id_rsa
source $HOME/.keychain/$HOSTNAME-sh
```

Source : http://www.cyberciti.biz/faq/ubuntu-debian-linux-server-install-keychain-apt-get-command

Navigation via SSH

Pour voir les fichiers avec konqueror et Nautilus rien de plus simple

Konqueror

```
fish://user@192.168.x.x
```

Nautilus

```
ssh://user@192.168.x.x:22
```

Tunnel chiffré en SSH

Il se peut que vous vouliez établir une connexion distante pour transiter des données de manière 100% transparente et sécurisée, nous allons donc établir un tunnel ssh.

```
ssh -L 5901:localhost:5900 user@80.80.80.80
```

Cette technique est très utile pour relier en local un bon nombre d'utilisation, comme sur kde distant, un serveur smtp personnel, une boite mail (pop ou imap) personnelle, un bon nombre d'utilisations ont recours à cette technique.

Détail sur la ligne de commande SSH :

- ssh : invoque le protocole
- -L invoque de la création d'un tunnel crypté

- 5901 : port coté local
- localhost : indique que l'accès se fera en local de manière transparente
- 5900 : port du service distant
- user@80.80.80.80 : indique avec quel user et sur quelle ip on va établir la connection.

Configuration

Par défaut, SSH n'a pas besoin de configuration. Cependant, pour se simplifier la vie ou pour les cas particuliers, il est possible de préciser préalablement quelques informations sur les connexions que l'on effectue régulièrement.

Cela se fait au moyen de l'édition du fichier de configuration `~/.ssh/config`[6] :

```
nano ~/.ssh/config
```

Si vous vous connectez régulièrement sur l'ordinateur de votre maman, d'IP `ordi.de.ma.maman` sous le nom d'utilisateur `gaston` , et sachant que son serveur ssh est configuré pour écouter sur le `port 444` , ajoutez ceci dans votre fichier de configuration :

```
Host maman
Hostname ordi.de.ma.maman
User gaston
Port 444
```

Et, pour vous connecter sur l'ordi de votre môman, il vous suffira maintenant de taper simplement la ligne suivante :

```
ssh maman
```

Dans le même fichier de configuration, vous pouvez ajouter plusieurs *alias* comme celui-ci.

```
Host ServeurA
IdentityFile ~/.ssh/cleA
Host ServeurB
IdentityFile ~/.ssh/cleB
```

Pour une liste de toutes les options disponibles, et il y en a… *une floppée* ! Tapez :

```
man ssh_config
```

Problèmes courants

Après réinstallation du serveur

Si le pc serveur est réinstallé, la clé unique l'identifiant est changée.
Le client va détecter le changement et soupçonner que quelqu'un d'autre essaye de se faire passer pour le serveur original et vous obtiendrez alors un message comme celui-ci :

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx.
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of this
message.
Offending key in /home/user/.ssh/known_hosts:9
RSA host key for 192.168.X.XX has changed and you have requested strict
checking.
Host key verification failed.
```

Ce message indique la ligne contenant l'empreinte de l'ancien serveur ici :

```
 Offending key in ~/.ssh/known_hosts:9
```

Dans cet exemple, il faut alors supprimer la 9ème ligne de son fichier `/home/user/.ssh/known_hosts`

Donc nous éditons[7] ce fichier en user[8] :

```
nano ~/.ssh/known_hosts
```

et nous supprimons cette 9ème ligne.

À notre prochain contact vers le serveur, après le *yes* d'acceptation habituelle, une nouvelle clé d'identification sera créée et tout ira pour le meilleur des mondes **ssh** possible.

Après réinstallation du client

- Lors de la création de clés asymétriques côté client

Rappel : il s'était créé une paire de clés, rangée par exemple dans le dossier proposé ~/.ssh/ :
- "~/.ssh/id_rsa"
- "~/.ssh/id_rsa.pub" (celle qu'on a envoyé au serveur, par exemple avec la commande `ssh-copy-id` ou en `scp`.

Donc si on veut, pour une raison ou un autre[9] changer cette pair de clé, il faut :

- **Côté serveur** : Éditer le fichier /etc/ssh/sshd_config pour vérifier ou remettre l'authentification par mot de passe est à `yes` : `PasswordAuthentication yes`[10];

1. recharger ssh : `service ssh start` ;
2. supprimer le fichier "**~/.ssh/authorized_keys**"

- **Côté client** : supprimer les fichiers "~/.ssh/id_rsa" et "~/.ssh/id_rsa" puisqu'on va les régénérer en se créant une nouvelle paire de clés.

⇒ On peut ensuite recommencer la procédure : se connecter une première fois au serveur (si ça coince ne pas hésiter à supprimer sur le client ~/.ssh/known_hosts si on ne l'a pas fait) ; puis côté client créer une paire de clé et l'envoyer au serveur).

- Si tout va bien pour se connecter, mais qu'on veut simplement changer la passphrase de sa clé privée (crée avec `ssh-keygen -t dsa`), la commande est (côté client) :

```
ssh-keygen -p
```

Démarrer ssh-agent automatiquement (Debian minimale)

Pour démarrer `ssh-agent` au démarrage de votre session si vous utilisez un système minimal:

```
mkdir -p ~/.config/systemd/user
```

```
cat << EOF > ~/.config/systemd/user/ssh-agent.service
[Unit]
Description=SSH key agent

[Service]
Type=forking
Environment=SSH_AUTH_SOCK=%t/ssh-agent.socket
ExecStart=/usr/bin/ssh-agent -a $SSH_AUTH_SOCK

[Install]
WantedBy=default.target
EOF
```

Ensuite, dans votre `.bash_profile` (ou autre `profile` selon votre interpréteur):

```
echo export SSH_AUTH_SOCK="$XDG_RUNTIME_DIR/ssh-agent.socket" >>
.bash_profile
source .bash_profile
```

Il vous reste ensuite à démarrer le service:

```
systemctl --user enable ssh-agent.service # pour le démarrer automatiquement
systemctl --user start ssh-agent.service # pour démarrer le service
immédiatement
```

TP

- Pour COPIER un fichier sécurisé vous devez vous servir de : SCP

- Pour MONTER les fichiers servez-vous de : SSHFS
- Utiliser ssh comme un serveur de fichiers : sftp installation et configuration

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

2)

mais ça, c'est ssh qui s'en charge via openssl

3)

user

4)

PAN !

5)

Si besoin, pour permettre à nouveau la connexion par clé DSA, la solution est par là : https://debian-facile.org/viewtopic.php?pid=233935#p233935

6)

à créer s'il n'existe pas déjà

7)

vim ou nano

8)

voir cette documentation interne :su

9)

par exemple on a réinstallé le système anciennement client et on n'arrive plus à se connecter à son serveur

10)

le temps de la procédure, après laquelle, il veut mieux remettre la valeur “no”

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/doc:reseau:ssh:client

Last update: **18/08/2023 21:27**