

0.1 SSH (Secure SHell) : Connexion sécurisée à un ordinateur distant

SSH est le nom d'un protocole réseau mais c'est aussi un ensemble de programmes utilisant ce protocole. SSH vous permet de vous connecter à une machine distante et de transférer des fichiers depuis celui-ci ou vers celle-ci. Comme son nom l'indique, SSH est sécurisé, ce qui signifie que la connexion entre le client et le serveur est encryptée. De plus, ssh utilise une double authentification client et serveur ce qui isole potentiellement chaque connexion d'autres utilisateurs indelicats. L'intérêt d'un tel protocole, face à telnet, rlogin ou ftp est évident, c'est pourquoi je vous le conseille vivement en remplacement des protocoles sus cités, quand vous le pouvez. Il existe 2 versions de SSH et plusieurs logiciels permettant de les utiliser, je vais détailler l'installation client et serveur, linux et Windows de openssh.

0.1.1 1. Installation de openssh

Nous allons voir dans cette partie comment installer le serveur et le client ssh : openssh. Pour rendre ce didacticiel un minimum générique, je vais appeler le dossier de configuration de SSH : `$SYSCONFDIR`.

0.1.2 1.1 Installation à partir des packages de sa distribution

- Les utilisateurs de Mandriva, tapez : `urpmi openssh openssh-clients openssh-server`. Votre `$SYSCONFDIR` a pour valeur `/etc/ssh`.
- Les utilisateurs de Fedora, tapez : `yum install openssh openssh-clients openssh-server`. Votre `$SYSCONFDIR` a pour valeur `/etc/ssh`.
- Les utilisateurs de Debian, tapez : `apt-get install ssh`. Votre `$SYSCONFDIR` a pour valeur `/etc/ssh`.
- Les utilisateurs de Slackware, vous disposez du package openssh suivant : `n/openssh-xxx.tgz`. Il vous suffit de l'installer par : `installpkg /où_est/openssh-xxx.tgz`. Votre `$SYSCONFDIR` a pour valeur `/etc/ssh`.

0.1.3 1.2 Installation à partir des sources

Dans cette partie je vais détailler l'installation à partir des sources. En effet, il s'agit d'un logiciel sensé garantir un minimum de sécurité, votre version contient certainement des failles, il est donc indispensable d'avoir la dernière version. Première chose à faire, désinstaller tout logiciel se rapportant à ssh sur votre distribution. En général il s'agit, de ssh ou de openssh. Nous allons installer openssh, qui supporte ssh 1.x et 2. Téléchargez la dernière version d'openssh ici ¹ et installez-la par :

```
$ tar -xvzf /où_est/openssh-xxxx.tar.gz
$ cd openssh-xxx/
$ ./configure --prefix=/opt/ssh --sysconfdir=/etc/ssh
$ make
$ su
Password
# make install
```

¹ <ftp://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/>

A ce stade ssh est installé. Votre `$SYSCONFDIR` a pour valeur `/etc/ssh`. Il ne vous reste plus qu'à le configurer :

```
# chmod 750 /opt/ssh/sbin/sshd
# chmod 755 /opt/ssh/bin/ssh-keygen /opt/ssh/bin/scp
# chmod 755 /opt/ssh/libexec/sftp-server
```

Vous devez créer un utilisateur sshd sans droit :

```
# useradd sshd -d / -s /bin/false
```

Vous devez ensuite configurer votre fichier `/etc/ssh/sshd_config`, j'en fournis une copie adaptée à notre configuration ici², vous pourrez l'adapter à vos goûts, mais il est fonctionnel tel quel. Maintenant vous devez générer les clés par :

```
# /opt/ssh/bin/ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_key -N ""
# /opt/ssh/bin/ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ""
# /opt/ssh/bin/ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key -N ""
```

répondez "y" à toutes les questions puis lancez le daemon sshd :

```
# /opt/ssh/sbin/sshd
```

SSHD est désormais lancé, il ne vous reste plus qu'à créer un script de lancement à l'image de ceux présents dans `/etc/rc.d/init.d/`, pour automatiser le lancement au démarrage de la machine. A titre personnel, je fournis un script³ pour Mandriva, Fedora, Debian et Slackware. Il est inspiré de celui de Slackware et est suffisamment générique pour fonctionner avec la plus part des distributions.

– Pour Fedora et Mandriva Placez ce script dans `/etc/rc.d/init.d/`, nommez le sshd et activez-le par :

```
# cd /
# chmod 755 /etc/rc.d/init.d/sshd
# ln -s /etc/rc.d/init.d/sshd /etc/rc.d/rc2.d/S55sshd
# ln -s /etc/rc.d/init.d/sshd /etc/rc.d/rc3.d/S55sshd
# ln -s /etc/rc.d/init.d/sshd /etc/rc.d/rc4.d/S55sshd
# ln -s /etc/rc.d/init.d/sshd /etc/rc.d/rc5.d/S55sshd
# ln -s /etc/rc.d/init.d/sshd /etc/rc.d/rc0.d/K25sshd
# ln -s /etc/rc.d/init.d/sshd /etc/rc.d/rc1.d/K25sshd
# ln -s /etc/rc.d/init.d/sshd /etc/rc.d/rc6.d/K25sshd
```

– Pour Debian Placez ce script dans `/etc/init.d/`, nommez-le sshd et activez-le par :

```
# cd /
# chmod 755 /etc/init.d/sshd
# ln -s /etc/init.d/sshd /etc/rc2.d/S55sshd
# ln -s /etc/init.d/sshd /etc/rc3.d/S55sshd
# ln -s /etc/init.d/sshd /etc/rc4.d/S55sshd
# ln -s /etc/init.d/sshd /etc/rc5.d/S55sshd
# ln -s /etc/init.d/sshd /etc/rc0.d/K25sshd
# ln -s /etc/init.d/sshd /etc/rc1.d/K25sshd
# ln -s /etc/init.d/sshd /etc/rc6.d/K25sshd
```

²http://file.trustonme.net/data/sshd_config

³<http://file.trustonme.net/data/rc.sshd>

- Pour Slackware : Placez le script dans /etc/rc.d/, nommez-le rc.sshd et activez-le par :

```
# cd /etc/rc.d/  
# chmod 755 rc.sshd
```

Vous pouvez bien-sûr utiliser le script natif de votre distribution, mais à vous de l'adapter. Pensez également à ajouter /opt/ssh/bin au PATH globale. Les détails de la manipulation sont vus ICI⁴.

0.1.4 1.3 Configuration du serveur SSH

Toute la configuration du serveur SSH est enregistrée dans le fichier \$SYSCONFDIR/sshd_config. Quand vous tapez (remplacez \$SYSCONFDIR par sa valeur) :

```
$ sed /#/d $SYSCONFDIR/sshd_config
```

Vous devez avoir au minimum :

```
Port 22  
HostKey $SYSCONFDIR/ssh_host_key  
HostKey $SYSCONFDIR/ssh_host_rsa_key  
HostKey $SYSCONFDIR/ssh_host_dsa_key  
PermitRootLogin yes  
X11Forwarding yes  
Subsystem sftp /chemin_vers/sftp-server
```

Cela signifie que les lignes précédentes sont non commentées dans le fichier \$SYSCONFDIR/sshd_config. Notez que :

PermitRootLogin yes

autorise l'utilisateur root à se connecter via SSH au serveur. Si vous ne souhaitez pas cela, remplacez cette ligne par "PermitRootLogin no".

X11Forwarding yes

autorise les clients de votre serveur à lancer des applications graphiques. Si vous ne souhaitez pas cela, remplacez cette ligne par "X11Forwarding no".

/chemin_vers/

est à remplacer par le véritable chemin vers sftp-server.

0.1.5 2. Client SSH Linux

0.1.6 2.1 Les logiciels clients

En installant openssh comme ci-dessus, vous installez également la version cliente d'ssh, qui est formée de 4 logiciels :

⁴<http://www.trustonme.net/didactels/132.html>

ssh

qui est un shell sécurisé (remplace telnet), vous vous connecté à une machine et vous utilisez le shell par défaut (ou un autre) comme si vous étiez en face de la machine.

scp

autorise la copie sécurisée du client vers le serveur (remplace rcp).

sftp

elle permet l'upload et le download sécurisés (semblable à la commande ftp en mode console).

slogin

qui fonctionne de manière analogue à rlogin.

0.1.7 2.2 Configuration des clients

Sur chaque machine désireuse d'être cliente SSH, il doit y avoir un fichier `$$SYSCONFDIR/ssh_config`. Qui doit contenir au minimum les lignes suivantes non commentées :

```
Host *
ForwardX11 yes
```

Pour vous en assurer, tapez (remplacez `$$SYSCONFDIR` par sa valeur) :

```
$ sed /#/d $$SYSCONFDIR/ssh_config
Host *
ForwardX11 yes
```

0.1.8 2.3 Encore plus de sécurité

Pour plus de sécurité chaque utilisateur désireux d'utiliser ssh en client devra générer localement une clé. Si vous ne le faites pas, n'importe qui ayant votre password Unix pourra se connecter en tant que vous sur votre PC. D'un autre côté si vous le faites, vous ne pourrez plus vous connecter à distance sur votre PC que depuis, un poste qui a été balisé au préalable. Vous pouvez générer la clé par :

```
$ ssh-keygen -t dsa
```

Il vous demandera le nom du fichier à créer, pressez simplement [enter] le choix par défaut est correct. Il vous demandera ensuite un passphrase, qui est en fait un mot de passe, n'hésitez pas blindez-le ! Mais essayé de vous en souvenir quand même ;-) Chaque utilisateur devra également copier le contenu de son `~/ .ssh/id_dsa .pub` dans le `~/ .ssh/authorized_key` de la machine à laquelle il souhaite se connecter.

0.1.9 3. Client Windows

Loin de moi, l'idée de développer sur les capacités ssh de Windows, je vous recommande de vous reporter à l'un des nombreux sites qui parlent du sujet et qui vous en diront bien plus que moi. Je signale néanmoins, l'existence d'un excellent client ssh Windows nommé Putty⁵ je l'ai testé et j'en pense le plus grand bien. Vous devrez télécharger, `putty.exe` qui fait office de shell sécurisé (ssh sous unix), `pscp.exe` pour la copie sécurisée (scp sous unix) et enfin, `psftp.exe` pour le ftp sécurisé (sftp sous Unix).

0.1.10 4. Connexion

Mettons que je souhaite, me connecter à l'adresse 192.168.0.6 en tant qu'utilisateur kernel, sur mon réseau local, il me suffit de taper :

```
$ ssh kernel@192.168.0.6
```

Ça marche aussi sur Internet, par adresse IP bien-sûr mais aussi par le nom du site, grâce aux DNS qui font correspondre une adresse IP à un nom :

```
$ ssh kernel@fr.yahoo.com
```

Si tant est que yahoo aie un serveur ssh bien-sûr. A la première connexion, il y'a échange de clé, acceptez celle du serveur par "yes" et hop ! vous êtes connecté au PC distant. Vous remarquerez que votre shell a changé d'aspect, vous pouvez effectuer toutes les commandes de votre choix, excepté lancer des outils graphiques. Pour cela rendez-vous au 6. A partir de la connexion actuelle pour vous déplacer dans un réseau, au lieu d'utiliser rlogin, faites plutôt :

```
$ slogin -l kernel machine3
```

Où kernel est votre login et machine3 la machine distante. Pour vous déconnecter c'est : logout.

0.1.11 5. Transfert de fichiers

Je suppose dans la suite que mon PC (chez moi) a pour adresse 192.168.0.9 et que le PC distant sur mon LAN a pour adresse 192.168.0.6 : La commande :

```
$ scp test.txt kernel@192.168.0.9
```

Permet de transférer, le fichier test.txt de 192.168.0.9 (local) vers 192.168.0.6 (distant). Mais y'a beaucoup mieux : le ftp sécurisé⁶.

0.1.12 6. Lancer des applications graphiques distantes

Quelle est la problématique ? Je suis chez moi et je ne peux pas me déplacer, je dois tester une application révolutionnaire et en faire un rapport pour demain. Cette application est dispo sur un PC distant auquel je suis relié par LAN. Ce PC est allumé et fonctionne sous Unix/Linux, et enfin j'ai eu la bonne idée d'installer Linux chez moi. Joie et bonheur car SSH permet de lancer des applications graphiques depuis un PC distant et qui s'afficheront sur mon PC à moi. Cependant, ne rêvez pas trop, la réactivité des applications est fortement dépendante de votre connexion, sur 56 K oubliez ! Sur ADSL

⁵[http://www.chiark.greenend.org.uk/~sim\\$sgtatham/putty/](http://www.chiark.greenend.org.uk/~sim$sgtatham/putty/)

⁶<http://www.truostonme.net/didactels/318.html>

pourquoi pas, mais de temps en temps, sur LAN (1 à 10 M) à consommer sans modérations ! Je suppose dans la suite que mon PC (chez moi) a pour adresse 192.168.0.9 et que le PC distant sur mon LAN a pour adresse 192.168.0.6. Cette méthode suppose bien-sûr que le fichier `$SYS.CONFDIR/sshd_config` du PC distant (192.168.0.6) ait la ligne : `X11Forwarding yes` décommentée et que le `$SYS.CONFDIR/ssh_config` du PC local (192.168.0.9) ait la ligne : `ForwardX11 yes` décommentée. Concrètement, Je souhaite lancer le logiciel `application_qui_roxor`, donc je tape :

```
$ ssh -X kernel@192.168.0.6 application_qui_roxor &
```

Voilà `application_qui_roxor` est lancé, grâce à l'option `-X` qui gère le display, il existe d'autres méthodes qui utilisent les capacités réseaux d'X, mais elles posent trop de problèmes soit de sécurité soit de commodité.

0.1.13 7. Faire des Tunnels

Bien souvent vous voulez sécuriser votre lien VNC ou pourquoi pas sortir de votre entreprise car seulement les ports 80 et 443 sont ouverts en sortie. Rien de plus simple il faut que votre machine ait un accès ssh ouvert sur l'extérieur. C'est à dire que vous puissiez sortir par l'un port supportant SSL (22 ou 443). Pour commencer, lancez un terminal. Maintenant préparez votre commande, dans la ligne de commande qui va suivre il suffira de remplacer 8000 par le numéro du port que vous voudrez en local, de modifier `www.zalteam.dyndns.org` par votre adresse IP ou votre nom de domaine, la Zal ;)TeAm est pas un proxy anonyme, de remplacer 3128 par le port ouvert sur la machine distante (Ici c'est le port du proxy squid). Enfin, notez que `toto@zalteam.dyndns.org` est l'identifiant utilisez pour vous connecter sur votre machine en ssh. La ligne de commande que je tape chez moi est :

```
ssh -N -T -L 8000 :www.zalteam.dyndns.org :3128 toto@zalteam.dyndns.org
```

C'est bien beau d'ouvrir des ports mais comment j'utilise mon tunnel maintenant ? Rien de plus simple : Vous lancez firefox, pour continuer avec squid. Vous allez dans les paramètres de configuration du proxy et vous les configurer en utilisant l'adresse 127.0.0.1, pour le port celui que vous avez ouvert, ici c'est le 8000 pour notre exemple. Cette utilisation s'adapte aussi bien à du pop, imap ... Petit conseil, pour les serveurs ISA utiliser le client pour réencapsuler vos trames.