SSH Secure SHell

Qu'est-ce que SSH ?

SSH signifie *Secure SHell*. C'est un protocole qui permet de faire des connexions sécurisées (i.e. 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.

Problématique :

Installer un serveur SSH permet aux utilisateurs d'accéder au système à distance, en rentrant leur login et leur mot de passe (ou avec un mécanisme de clés). Cela signifie aussi qu'un pirate peut essayer d'avoir un compte sur le système (pour accéder à des fichiers sur le système ou pour utiliser le système comme une passerelle pour attaquer d'autres systèmes) en essayant plein de mots de passes différents pour un même login (il peut le faire de manière automatique en s'aidant d'un dictionnaire électronique). On appelle ça une attaque *en force brute*.

Il y a donc trois contraintes majeures pour garder un système sécurisé après avoir installé un serveur SSH :

- avoir un serveur SSH à jour au niveau de la sécurité, ce qui doit être le cas si vous faites consciencieusement les mises à jour de sécurité
- que les mots de passes de *TOUS* les utilisateurs soient suffisamment complexes pour résister à une attaque en force brute ;
- surveiller les connexions en lisant régulièrement le fichier de log `/var/log/auth.log`.

Le système de clés de SSH :

La théorie de la cryptographie asymétrique :

SSH utilise la cryptographie asymétrique RSA ou DSA. En cryptographie asymétrique, chaque personne dispose d'un couple de clé : une clé publique et une clé privée. La clé publique peut être librement publiée tandis que la clé privée doit rester secrète. La connaissance de la clé publique ne permet pas d'en déduire la clé privée.

Si la personne A veut envoyer un message confidentiel à la personne B, A crypte le message avec la clé publique de B et l'envoie à B sur un canal qui n'est pas forcément sécurisé. Seul B pourra décrypter le message en utilisant sa clé privée.

La théorie de la cryptographie symétrique :

SSH utilise également la cryptographie symétrique. Son principe est simple : si A veut envoyer un message confidentiel à B, A et B doivent d'abord posséder une même clé secrète. A crypte le message avec la clé secrète et l'envoie à B sur un canal qui n'est pas forcément sécurisé. B décrypte le message grâce à la clé secrète. Toute autre personne en possession de la clé secrète peut décrypter le message.

L'établissement d'une connexion SSH :

Un serveur SSH dispose d'un couple de clés RSA stocké dans le répertoire `/etc/ssh/` et généré lors de l'installation du serveur. Le fichier `ssh_host_rsa_key` contient la clé privée et

a les permissions 600. Le fichier `ssh_host_rsa_key.pub` contient la clé publique et a les permissions 644.

Nous allons suivre par étapes l'établissement d'une connexion SSH :

a. Le serveur envoie sa clé publique au client.
b. Le client génère une clé secrète et l'envoie au serveur, en cryptant l'échange avec la clé publique du serveur (cryptographique asymétrique). Le serveur décrypte la clé secrète en utilisant sa clé privée, ce qui prouve qu'il est bien le vrai serveur.
c. Pour le prouver au client, il crypte un message standard avec la clé secrète et l'envoie au client. Si le client retrouve le message standard en utilisant la clé secrète, il a la preuve que le serveur est bien le vrai serveur.
d. Une fois la clé secrète échangée, le client et le serveur peuvent alors établir un canal sécurisé grâce à la clé secrète commune (cryptographie symétrique).
e. Une fois que le canal sécurisé est en place, le client va pouvoir envoyer au serveur le login et le mot de passe de l'utilisateur pour vérification. La canal sécurisé reste en place jusqu'à ce que l'utilisateur se déloggue.

La seule contrainte est de s'assurer que la clé publique présentée par le serveur est bien sa clé publique... sinon le client risque de se connecter à un faux serveur qui aurait pris l'adresse IP du vrai serveur (ou toute autre magouille). Une bonne méthode est par exemple de demander à l'administrateur du serveur quelle est le *fingerprint* de la clé publique du serveur avant de s'y connecter pour la première fois. Le *fingerprint* d'une clé publique est une chaîne de 32 caractères hexadécimaux unique pour chaque clé ; il s'obtient grâce à la commande **ssh-keygen -l**.

Installation et configuration de SSH :

Installation du client et du serveur SSH :

Le client et le serveur SSH sont dans le même package *ssh*. Ce package est installé dès la première utilisation de *dselect*. Si vous avez bien respecté nos consignes lors de la procédure d'installation . Vous n'avez pas activé le serveur SSH.
Maintenant que votre système est à jour niveau sécurité, vous pouvez activer le serveur SSH, si vous le souhaitez. Pour cela, supprimez le fichier `/etc/ssh/sshd_not_to_be_run` et lancer SSH :

```
# rm /etc/ssh/sshd_not_to_be_run
# /etc/init.d/ssh start
Starting OpenBSD Secure Shell server: sshd.
```

Configuration du serveur SSH :

Le fichier de configuration du serveur SSH est `/etc/ssh/sshd_config`. A ne pas confondre avec le fichier `/etc/ssh/ssh_config`, qui est le fichier de configuration du client SSH.
Nous allons vous commenter les lignes les plus importantes de ce fichier de configuration :

```
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. Seuls certains vieux clients SSH ne

savent faire que du 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 vous logguer en root par SSH. Vous pouvez changer et mettre "*no*", ce qui signifie que pour vous connecter en root à distance, vous devrez d'abord vous connecter par SSH en tant que simple utilisateur, puis utiliser la commande **su** pour devenir root. C'est une sorte de double protection.

```
X11Forwarding yes
```

Signifie que vous allez pouvoir travailler en export display par SSH.

Si vous avez modifié le fichier de configuration du serveur, il faut lui dire de relire son fichier de configuration :

```
# /etc/init.d/ssh reload
Reloading OpenBSD Secure Shell server's configuration.
```

- **Création de paire de clefs**

Ssh s'appuie sur des algorithmes à paire de clefs, ce qui signifie que vous disposez d'une clef publique, disponible pour tout un chacun et une clef privée dont vous gardez jalousement l'entrée. Ce système va nous permettre de nous identifier auprès des hôtes que nous désirons contacter. Il nous faut au préalable créer le trousseau.

```
$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/jop/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jop/.ssh/id_dsa.
Your public key has been saved in /home/jop/.ssh/id_dsa.pub.
The key fingerprint is:
4a:0b:3b:eb:ed:05:47:56:cb:23:28:d3:d7:81:69:08

$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jop/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jop/.ssh/id_rsa.
Your public key has been saved in /home/jop/.ssh/id_rsa.pub.
The key fingerprint is:
52:65:28:9a:8b:64:cb:b7:6e:70:75:10:d9:0a:01:d9
```

Pour les deux algorithmes (dsa, rsa), le système nous demande dans quel fichier nous désirons sauvegarder la clef. Les fichiers par défaut semblent une bonne solution. 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.

Connexion SSH :

Nous allons maintenant voir **trois méthodes de connexion via ssh**.

- **Connexion par mot de passe :**

La première des méthodes, la plus connue et la plus utilisée, reposant sur le modèle employé par rlogin ou rsh; l'hôte distant demande un mot de passe pour s'assurer de votre identité.

```
$ ssh -p 222 -l root 192.168.0.1
The authenticity of host '192.168.0.1 (192.168.0.1)' can't be established.
RSA key fingerprint is 74:4c:57:fd:c2:2c:0d:c3:3f:09:01:7d:e8:b7:21:24.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.1' (RSA) to the list of known hosts.
root@192.168.0.1's password:
Last login: Thu Dec 26 03:37:03 2002 from centaur
$
```

Je me connecte au port 222 sur une machine de mon réseau local, demandant de m'identifier comme root. La machine n'est pas connue dans mon fichier **known_hosts**. Ce fichier contient les clés hôte DSA des serveurs SSH auxquels l'utilisateur s'est connecté. Cette méthode de connexion est intéressante, mais minimise les capacités que vous avez avec ssh. Pour des connexions telles que vers un serveur CVS, un tunnel pour votre courrier, il serait fastidieux de s'authentifier à chaque fois par ce moyen.

- **Connexion par paires de clef :**

Puisque nous utilisons des algorithmes à paire de clefs (rsa, dsa), c'est à dire composée d'une clef secrète et d'une clef publique, il faut bien que cela nous serve à quelque chose. Nous allons donc automatiser la connexion. Pour ce faire, votre hôte contient un fichier **authorized_keys** dans le répertoire **.ssh** où vous vous connectez (en général un home directory).Il suffit de copier l'identifiant ou les identifiants de vos clefs publiques pour que vous soyez reconnu du serveur.
Attention, il faut que *PubkeyAuthentication* soit positionné à *yes* dans vos fichiers de configuration.
Pour insérer ma clef, j'ai plusieurs méthodes à ma disposition. Je peux employer des moyens conventionnels tels que le ftp ou le mail (à l'administrateur par exemple), ou je peux le faire au moyen d'outil sécurisé. Puisque c'est notre sujet, profitons en pour donner un exemple de *scp* sur lequel nous reviendrons ultérieurement.

```
$scp .ssh/id_dsa.pub jop@scipc-jpg:/home/jop/.ssh/dsa2connex
Warning: Permanently added 'scipc-jpg' (RSA) to the list of known hosts.
jop@scipc-jpg's password:
id_dsa.pub 100% |*****************************| 613 00:00
Mon fichier est maintenant copié sur l'hôte distant, il me reste à inclure
la clef dant le fichier authorized_keys. Une simple commande suffira :

$ cat dsa2connex >>authorized_keys
```

Je peux maintenant me connecter, l'hôte distant me reconnaît. Je peux me connecter sans mot de passe et avec une passphrase si j'en ai désiré une.

- **L'agent d'authentification**

Nous savons maintenant nous connecter à un hôte distant connaîssant notre identité par l'intermédiaire de notre clef publique. Nous avons vu précédement que nous étions libre de mettre ou non une passphrase afin de chiffrer notre identification et compliquer l'usurpation qui pourrait en être faite [mode parano on]. S'il paraît fastidieux d'insérer un mot de passe à chaque connexion, cela l'est d'autant plus lorsque l'on doit rentrer une phrase entière. L'agent ssh est là pour nous simplifier la vie. Lancé au début de la session (terminal ou graphique), il retient la passphrase le temps où **ssh-agent** sera actif (le temps d'une session).

Pour une session en mode terminal :
```
$ssh-agent screen
$ssh-add
```

Après avoir lancé l'agent ssh, on ajoute la passphrase à l'agent par l'intermédiaire de **ssh-add** ; tous les hôtes distants disposant de votre clef publique ne vous demanderons alors plus la passphrase puisque gérée par l'agent ssh.

De même, en mode graphique :
```
$ssh-agent startx
```
Il vous suffit ensuite d'ouvrir un terminal et de taper :
```
$ssh-add
```
L'agent sera actif pour toutes les applications utilisées en mode graphique.

Transférer un fichier avec SCP :

scp (secure copy program) est un binaire fournit avec SSH, il permet de publier ou de télécharger des fichiers d'une façon sécurisée et cryptée entre votre serveur local ou un ou plusieurs serveur distant. Il utilise SSH pour transférer les fichiers et utilise le même système d'authentification que SSH.

Pour copier des fichiers ou un répertoire depuis votre serveur local vers un serveur distant :
scp fichier1 fichier2 fichier3 utilisateur@serveurdistant:/repertoiredistant/
scp -r repertoire utilisateur@serveurdistant:/repertoiredistant/

Bien sûr, si vous n'utilisez pas l'authentification SSH par clé publique, vous devrez fournir un mot de passe pour publier ces fichiers ou ce répertoire.

Pour copier un fichier ou un répertoire depuis un serveur distant vers votre serveur local
scp utilisateur@serveurdistant:/repertoiredistant/fichier1 /repertoirelocal/
scp -r utilisateur@serveurdistant:/repertoiredistant/ /repertoirelocal/

Pour copier un fichier d'un premier serveur distant vers un second serveur distant à partir de votre serveur local :
scp utilisateur@serveurdistant1:/repertoiredistant1/fichier
utilisateur@serveurdistant2:/repertoiredistant2/
scp -r utilisateur@serveurdistant1:/repertoiredistant1/
utilisateur@serveurdistant2:/repertoiredistant2/

Attention dans ce cas, il vous faut obligatoirement avoir une authentification SSH par clé publique entre le serveur distant1 et le serveur distant2

scp propose quelques options comme :
-r : copie récursivement le contenu d'un dossier.
-v : permet à scp d'être un peu plus bavard.
-p : préserve les dates de modifications, les dates d'accès, les droits du fichier original
-q : désactive la barre de progression de téléchargement.
-P : permet de spécifier le port sur lequel on veut se connecter.

Bibliographie :

Linux Administration de « Jean-François Bouchaudy » et « Gilles Goubet »
http://lea-linux.org
http://people.via.ecp.fr/~alexis/formation-linux/ssh.html