Connexion sécurisée grâce à SSH

Régis Senet

Protégez vos communications de l'ensemble des actes de piratage en chiffrant vos données ! La mise en place de protocole sécurisé pour les communications distantes est vivement recommandée du fait que nous ne pouvons pas savoir qui nous écoute à chaque instant dans l'immensité de l'internet. Il est donc temps de remplacer tous ces protocoles laissant vos données transiter en claires sur le réseau, et de posséder vos accès SSH.

SSH ou bien Secure Shell est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Par la suite toutes les trames sont chiffrées. Il devient donc impossible d'utiliser un sniffer tel que Wireshark pour voir ce que fait l'utilisateur via les données qu'il reçoit ou envoi. Le protocole SSH a initialement été conçu avec l'objectif de remplacer les différents programmes rlogin, telnet et rsh qui ont la fâcheuse tendance de faire passer en clair l'ensemble des données d'un utilisateur vers un serveur et inversement, permettant à une personne tierce de récupérer les couples de login/mot de passe, les données bancaire etc.

Le protocole SSH existe en deux versions: la version 1.0 et la version 2.0. La version 1.0 souffrait de failles de sécurité et fut donc rapidement rendue obsolète avec l'apparition de la version 2.0. La version 2 est largement utilisée à travers le monde par une grande majorité des entreprises. Cette version a réglé les problèmes de sécurité liée à la version 1.0 tout en rajoutant de nouvelles fonctionnalités telles qu'un protocole de transfert de fichiers complet. La version 1.0 de SSH a été conçue par Tatu Ylönen, à Espoo, en Finlande en 1995. Il a créé le premier programme utilisant ce protocole et a ensuite ouvert une société, SSH Communications Security pour exploiter cette innovation. Cette première version utilisait certains logiciels libres comme la bibliothèque Gnu libgmp, mais au fil du temps ces logiciels ont été remplacés par des logiciels propriétaires. SSH Communications Security a vendu sa licence SSH à F-Secure.

Ce qu'il faut savoir...

Connaissance en systeme d'exploitation UNIX/Linux.

Cet article explique...

- L'intérêt d'utiliser des protocoles sécurisé,
- La mise en place d'un serveur SSH.

Installation

Au cours de cet article, la distribution utilisée fut une Debian 5.0 (*Lenny*) entièrement mise à jour. Attention, il est possible que certaines commandes ne soient pas tout à fait identiques sur une autre distribution. L'ensemble des installations va se réaliser grâce au gestionnaire de paquets propre à un système Debian : APT (Advanced Package Tool).

Mise à jour du système

Il est possible à tout moment qu'une faille de sécurité soit découverte dans l'un des modules composant votre système que ce soit Apache ou quoi que ce soit d'autre. Certaines de ces failles peuvent être critiques d'un point de vue sécurité pour l'entreprise. Afin de combler ce risque potentiel, il est nécessaire de régulièrement mettre à jour l'ensemble du système grâce à divers patches de sécurité.

Il est possible de mettre à jour l'ensemble du système via la commande suivante :

```
nocrash:~# apt-get update && apt-get
upgrade
```

Le système d'exploitation est maintenant complètement à jour, il est donc possible de mettre en place un serveur SSH dans de bonnes conditions. Il est possible de ne pas passer par cette étape mais elle est fortement conseillée pour la sécurité ainsi que la stabilité de votre système d'exploitation.

Installation de SSH

Dans un premier temps, nous allons réaliser l'installation via le gestionnaire de paquet propre à un système Debian, le système APT (*Advanced Package Tool*). Nous verrons l'installation via les sources un peu plus tard :

```
nocrash:~# apt-get install ssh
```

Installation de SSH en ligne de commande.

- Les paquets suivants sont des dépendances du paquet SSH :
  `openssh-client`
  client shell sécurisé
  `openssh-server`
- Serveur shell sécuritaire

Installation via les sources

Nous allons à présent voir l'installation via les sources directement disponibles sur le site officiel d'OpenSSH (*http://www.openssh.org/*).

Dans l'éventualité où vous avez déjà installé OpenSSH via le gestionnaire de paquet comme vu précédemment, il est nécessaire de le désinstaller avant de faire une nouvelle installation :

```
nocrash:~# apt-get autoremove ssh
```

Une fois correctement désinstallé, il est possible de réaliser l'installation par les sources :

```
nocrash:~# mkdir /usr/ssh
nocrash:~# cd /usr/ssh/
nocrash:~# wget ftp://
ftp.openbsd.org/pub/OpenBSD/OpenSSH/
portable/openssh-5.2p1.tar.gz
nocrash:~# tar xzvf openssh-
5.2p1.tar.gz
nocrash:~# cd openssh-5.2p1/
nocrash:~# ./configure --bindir=
/usr/local/bin --sbindir=
/usr/sbin--sysconfdir=/etc/ssh
nocrash:~# make && make install
```

En cas d'erreur, reportez-vous à NB.

Configurations préalable

Sur certaines distribution, afin de pouvoir activer le serveur SSH, il est nécessaire de supprimer un fichier présent par défaut, le fichier */etc/ssh/sshd_not_to_be_run* avant de pouvoir lancer le serveur SSH.

```
nocrash:~# rm -rf
   /etc/ssh/sshd_not_to_be_run
```

Une fois le fichier supprimé (s'il existe), il est possible de passer à la phase de configuration.

Configuration du serveur SSH

Le fichier regroupant l'ensemble des configurations du serveur SSH se trouve être le fichier */etc/ssh/sshd_config*. Nous allons éditer ce fichier afin de pouvoir y faire nos modifications :

```
nocrash:~# vi /etc/ssh/sshd_config
```

Voici les paramètres principaux au bon paramétrage de notre serveur SSH :

```
Port 22
```

Cette directive signifie simplement que le serveur SSH va écouter sur le port 22 (port par défaut). Il est possible de faire écouter le serveur SSH sur plusieurs ports en rajoutant plusieurs fois cette directive avec des ports différents :

```
Protocol 2
```

Cette directive permet de spécifier que seul la version 2 du protocole SSH sera utilisée, la version 1 de SSH est obsolète pour des raisons de sécurité :

```
PermitRootLogin no
```

Cette directive permet d'empêcher toute connexion à distante avec l'utilisateur root (superutilisateur). Un accès distant grâce avec l'utilisateur root par une personne mal intentionnée pourrait être catastrophique pour l'intégrité du système :

```
PermitEmptyPasswords no
```

Cette directive permet d'interdire les connexions avec un mot de passe vide, il est indispensable de mettre cette directive à *no* :

```
LoginGraceTime 30
```

Cette directive permet de limiter le laps de temps permettant de se connecter au SSH

```
AllowUsers nocrash
AllowGroups admin
```

Ces directives donnent la possibilité de n'autoriser que certains utilisateur et/ou groupe :

XXX

NB. Si vous système a récemment été installé, il est possible que certaine librairies soit manquante. Il est nécessaire de les installer :

- Librairie `gcc` : `apt-get install gcc`
- Librairie `libcrypto / SSL` :

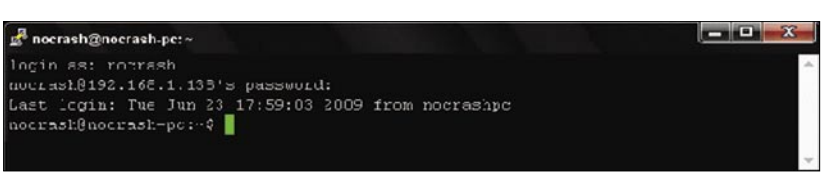

Figure 2. Nous voici ainsi connecté sur notre serveur distant grâce à Putty

Figure 3. puTTYKey generator

```
AllowTcpForwarding no
X11Forwarding no
```

Ces directives permettent de désactiver le transfert de port TCP et le transfert X11.

Authentification

Il existe deux méthodes afin de pouvoir s'authentifier en SSH sur une machine distante. Ces deux méthodes sont :

- Authentification par clé,
- Authentification par mot de passe.

Authentification par clé

L'authentification par clé est un très bon moyen pour s'authentifier de manière sécurisé. En effet, des clés asymétriques de type DSA vont être générées.

Afin de générer nos clés DSA, nous allons utiliser la commande suivante :

```
nocrash:~# ssh-keygen -t dsa
```

Les clés générées par défaut auront une taille de 1024 bits, ce qui est largement suffisant pour assurer une bonne protection.

Deux clés vont donc être générées :

- Une clé publique présente dans le fichier *~/.ssh/id_dsa.pub* avec les permissions 644.
- Une clé privée présente dans le fichier *~/.ssh/id_dsa* avec les permissions 600.

Lors de la création des clés, une passphrase vous sera demandée, il est important de choisir un mot de passe complexe. En effet, cette passphrase permet de crypter la clé privée (clé devant rester absolument à votre seule connaissance).

Au cas où vous vous seriez trompé dans votre passphrase, il est toujours possible de la modifier grâce à la commande suivante :

```
nocrash:~# ssh-keygen -p
```

La dernière étape avant de pouvoir s'authentifier par clé publique est d'autoriser sa propre clé. Pour cela, il est nécessaire d'ajouter votre clé publique dans le fichier /.ssh/authorized_keys de la machine sur laquelle, vous voulez vous connecter.

Pour réaliser cela, il est nécessaire d'utiliser la commande suivante :

```
nocrash:~# ssh-copy-id -i ~/.ssh/id_
dsa.pub login@Adresse_de_la_machine
```

Pour mon exemple :

```
nocrash:~# ssh-copy-id -i ~/.ssh/id_
dsa.pub nocrash@192.168.1.138
```

Voici à quoi pourrait ressembler le fichier de configuration :

```
Port 22
Protocol 2
ListenAddress 192.168.1.138
ServerKeyBits 1024
PermitRootLogin no
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/
authorized_keys
IgnoreRhosts yes
#(mettez ces 2 options à no si vous
ne voulez offrir l'accès qu'aux
utilisateurs ayant enregistré leur
clés):
```

Authentification par mot de passe

L'authentification par mot de passe quand à elle est une authentification très simple à mettre en place du fait qu'il n'y a rien à mettre en place.

Il est nécessaire que l'utilisateur voulant se connecter possède un compte sur la machine distante et c'est tout.

Dans l'exemple suivant, nous voulons nous connecter avec l'utilisateur *NoCrash* sur la machine répondant à l'adresse IP 192.168.1.138. Nous allons donc vérifier que cet utilisateur existe bien avant tout. Dans le cas où il n'existerait pas, il est nécessaire de le créer sur la machine distante avec un mot de passe (ne pas oublier la directive PermitEmptyPasswords no).

```
nocrash:~# cat /etc/passwd | grep
nocrash
nocrash:x:1000:1000:nocrash,,,:
/home/nocrash:/bin/bash
```


A propos de l'auteur...

Régis SENET est actuellement étudiant en quatrième année à l'école Supérieur d'informatique Supinfo. Passionné par les tests d'intrusion et les vulnérabilités Web, il tente de découvrir la sécurité informatique d'un point de vue entreprise. Il est actuellement en train de s'orienter vers le cursus CEH, LPT et Offensive Security.
Contact : *regis.senet@supinfo.com*
Site internet : *http://www.regis-senet.fr*
Page d'accueil : *http://www.openssh.com/*

Le *x* juste après le login permet de spécifier qu'un mot de passe est bien présent.

Voici à quoi pourrait ressembler le fichier de configuration :

```
Port 22
Protocol 2
ListenAddress 192.168.1.138
ServerKeyBits 1024
PermitRootLogin no
PubkeyAuthentication no
IgnoreRhosts yes
PasswordAuthentication yes
Compression yes
```

A présent que l'ensemble des configurations sont faites, il est nécessaire de lancer le serveur SSH. Pour cela, nous allons utiliser la commande suivante :

```
nocrash:~# /etc/init.d/ssh start
```

Si tout ce passe bien, nous allons avoir le message suivant :

```
Starting OpenBSD Secure Shell
server :sshd.
```

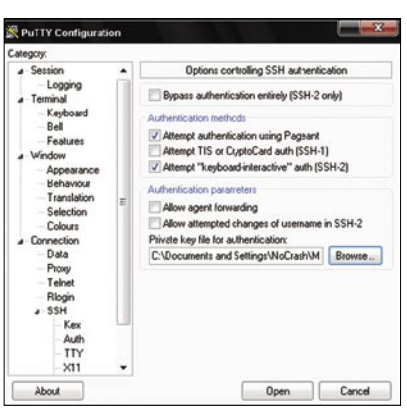

Figure 4. Configuration de Putty

Il est alors possible de pouvoir se connecter à la machine distante.

Connexion

Afin de se connecter en SSH sur une machine distante possédant un serveur SSH, il est possible d'utiliser la ligne de commande dans le cas où vous êtes sous Linux ou MAC.

Pour cela, voici la commande à utiliser :

```
nocrash:~# ssh login@
   Adresse_de_la_machine
```

Soit pour notre exemple :

```
nocrash:~# ssh
nocrash@192.169.1.138
```

Pour les machines de type Windows, il n'existe pas de client SSH en natif, il est alors néces saire de passer par des logiciels, tels que Putty.

Authentification par clé

Comme il est possible de le voir dans l'authentification par mot de passe, nous allons tenter de nous connecter au serveur distant via SSH grâce à Putty.

Putty ne sachant pas gérer les clefs générées par le serveur avec ssh-keygen il faut utiliser l'utilitaire puttygen afin de générer un couple de clés utilisable. Ouvrez puTTYKey generator puis générer une nouvelle paire de clés.

Cliquez à présent sur *Save public key* et *Save private key* afin de sauvegarder les clés. Il est à présent nécessaire de copier la clé publique que vous venez de générer sur le serveur distant à l'adresse suivante : *~/.ssh/authorized_keys*

Une fois les clés générées, il est possible de se connecter avec Putty. Il est nécessaire de spécifier l'emplacement de la clé privée dans *Connection>>SSH>>Auth* avant de se connecter.

Astuces

Dans le fichier de configuration de ssh soit le fichier */etc/ssh/sshd_config*, il n'est pas obligatoire mais fortement conseillé de modifier le port utilisé par SSH. Par défaut, il s'agit du port 22. Ce petit changement permet de *brouiller* un attaquant qui s'attendrais à voir un SSH sur le port 22 et non pas sur le port 1998 par exemple :

```
Port 1998
```

Afin de vérifier que vos mots de passe sont assez complexes, il est possible de tenter de les casser vous-mêmes afin de vérifier si quelqu'un d'autre en est capable. Pour cela, il est nécessaire d'installer John The Ripper, un utilitaire de cassage de mot de passe :

```
nocrash:~# apt-get install john
```

Il est maintenant possible de vérifier vos mots de passe :

```
nocrash:~# john /etc/shadow
```

Les mots de passe trouvés sont donc jugés comme étant trop simple, il est préférable de les modifier.

Conclusion

La mise en place de protocole sécurisé pour les communications distantes est vivement recommandé du fait que nous ne pouvons pas savoir qui nous écoute à chaque instant dans l'immensité de l'internet. Il ne faut pas non plus croire que le danger vient simplement de l'internet. En effet, la majorité des actes de piratages informatique se font au sein d'une même entreprise par les employés eux-mêmes. Il est donc temps de protéger vos communications de l'ensemble de ces voyeurs, il est temps de chiffrer vos données, il est temps de remplacer tous ces protocoles laissant vos données transiter en claires sur le réseau, il est temps de posséder vos accès SSH.