OFPPT مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

Complexe de Formation dans les Métiers des Nouvelles Technologies de l'Information, de l'Offshoring et de l'Electronique -Oujda

Module : Administration d'un réseau

OPENSSH

Formatrice : ZITI Ilham

Sommaire

1. Présentation SSH

Secure Shell (SSH) est un programme mais aussi un protocole de communication sécurisé. Grâce à SSH, on peut se connecter à distance sur une machine, transférer des fichiers, si celle-ci dispose d'un serveur SSH,

SSH remplace de manière sécurisée :

- Telnet: Vous pouvez exécuter des commandes depuis un Réseau Local ou Internet via SSH
- FTP: Si vous ne souhaitez qu'ajouter ou modifier des fichiers sur un serveur, SSH est bien plus adapté que FTP
- Et d'autres, via le tunneling: SSH permet d'accéder à un service réseau en le faisant circuler dans SSH pour profiter de toutes les protections qu'il apporte. Vous pouvez donc sécuriser n'importe quel protocole grâce à SSH, comme VNC par exemple.

SSH permet de faire, en usage de base :

- Accès à distance sur la console en ligne commande (shell), ce qui permet, entre autres, d'effectuer la totalité des opérations courantes et/ou d'administration sur la machine distante.
- Déporter l'affichage graphique de la machine distante.
- Transferts de fichiers en ligne de commande.
- Montage ponctuel de répertoire distant, soit en ligne de commande, soit via Nautilus, sous Gnome par exemple.
- Montage automatique de répertoires distants

2. Présentation Openssh

OpenSSH (OpenBSD Secure Shell) est un ensemble d'outils informatiques libres permettant des communications sécurisées sur un réseau informatique en utilisant le protocole SSH.
OpenSSH chiffre tout le trafic (mots de passe y compris), via une combinaison astucieuse de chiffrement symétrique et asymétrique. OpenSSH fournit également d'autres méthodes d'authentification alternatives au traditionnel mot de passe. En ce qui concerne l'authentification, il supporte le RSA et le DSA.

3. Les principaux composants de SSH :

sshd : le logiciel serveur, actif sur le port 22, qui ouvre une session à partir d'une connexion d'un client ssh.

ssh : le logiciel client qui remplace rsh et rlogin.
scp : le logiciel client qui remplace rcp.
ssh-keygen : le logiciel qui permet de créer un couple de clés publique/privée

4. Installation du serveur OpenSSH

Pour installer le serveur ssh sur notre machine, installer le paquet openssh-server

Au début nous allons est ce que le serveur est déja installer en utilisant la commande :

```
[root@localhost ofppt]# rpm -qa openssh-server
```

Si le serveur n'est pas installer, la commande permetant de l'installer est la suivante:

```
[root@localhost ofppt]# yum install openssh-server
```

Lancer le service au démarrage

```
# systemctl enable sshd.service
```

5. Configurer le service SSH

Editez le fichier /etc/ssh/sshd_config. Le fichier contient un ensemble de directives toutes commentées. En fait celles-ci nous informent sur les valeurs par défaut, il ne faut décommenter l'une d'entre elles que lorsque nous souhaitons en changer la valeur.

NB : - Il faut être root pour modifier la configuration du serveur
- Dans le fichier de configuration original, toutes les valeurs commentées (précédées d'un #) sont placées à leur valeur par défaut.

5.1 Désactiver les connexions SSH en root

Il est fortement conseillé pour des raisons de sécurité d'interdire la connection au serveur openSSH avec le login root. Pour cela, éditer le fichier /etc/ssh/sshd_config et modifier cette ligne:

```
PermitRootLogin no
```

Les valeurs possibles sont donc yes pour autoriser l'accés au compte root par SSH, no (par défaut) pour la refuser

5.2 Autoriser/Interdire des utilisateurs

Pour autoriser une liste de certains utilisateurs à se connecter. Modifier ou ajouter cette ligne dans le sshd_config :

```
AllowUsers user1 user2 user3
```

Pour autoriser seulement certains membres de groupes à avoir accès via SSH en modifiant la ligne :

```
AllowGroups goupe1 groupe2
```

Pour refuser la connexion que de certains utilisateurs. Modifier ou ajouter cette ligne dans le sshd_config :

```
DenyUsers user1 user2 user3
```

5.3 Afficher une bannière

Si le serveur est public, on peut ajouter une bannière, une sorte de message affiché à la connexion.

Exemple :

```
#############################################################################
#                                                                           #
#  WARNING : Unauthorized access to this system is forbidden and will be    #
#  prosecuted by law. By accessing this system, you agree that your actions #
#  may be monitored if unauthorized usage is suspected.                     #
#                                                                           #
#############################################################################
```

Décommenter la ligne "Banner" dans le fichier /etc/ssh/sshd_config et on y place le nom du fichier de la bannière :

```
Banner /etc/banner
```

Puis créer le fichier /etc/banner avec notre bannière dedans

5.4 Modifier le port d'écoute

Par défaut, le serveur openSSH écoute sur le port 22, pour change le port d'écoute du serveur openSSH modifier ou ajouter cette ligne dans le sshd_config :

```
Port  numéro_du_port
```

5.5 Modifier l'interface d'écoute

```
ListenAddress 0.0.0.0
```

Cette option permet de faire écouter le démon du serveur OpenSSH, sshd, que sur une interface donnée.
Si par exemple vous possédez 2 cartes réseaux, une permettant une connexion sur le réseau internet (avec une adresse non locale) et une autre permettant une connexion locale (par exemple avec une adresse 192.168.0.1), et que vous souhaitez que les accés SSH ne se fassent qu'en provenance du réseau local entrez votre IP locale.Dans ce cas, les adresses extérieurs à votre réseau locale n'auront pas accés à votre serveur OpenSSH.

5.6 Autorier /interdir mot de passe vide :

Pour interdir la connexion au mot de passe vide modifier ou ajouter cette ligne dans le sshd_config :

```
PermitEmptyPasswords no
```

Les valeurs possibles sont donc **yes** pour autoriser l'accés mot de passe vide , **no** (par défaut) pour la refuser

5.7 Autoriser interdire authentification par mot depasse

L'option suivante permet d'autoriser ou non des connexion avec un couple identifiant/mot de passe

```
PasswordAuthentication yes
```

Les valeurs possibles sont donc yes pour autoriser l'authetification par mot de passe , no pour la refuser

5.8 Limiter le nombre de connexion

L'option suivante permet de spécifier le nombre maximal de sessions ouvertes autorisées par connexion réseau. La valeur par défaut est 10.

```
MaxSessions 10
```

L'option suivante permet de spécifier le nombre maximal de connexions non authentifiées simultanées au démon SSH La valeur par défaut est 10.

```
MaxStartups 10
```

```
MaxStartups 10:30:60
```

Cette option permet de limiter le nombre de connexion : le 10 représente le nombre de connexions acceptées sans qu'un utilisateur ait réussi à s'identifier, si cela passe au dessus de 10, il y a 30 % de chances que les suivantes soient bloquées, ça augmente linéairement jusqu'à complétement bloquer les connexions à 60 %

5.9 Limiter le nombre de tentative d'authentification

Pour limiter le nombre de de tentative d'authentification par exemple à 4 , modifier la ligne suivante :

```
MaxAuthTries 4
```

5.10 Maintenir la connexion

L'option ClientAliveInterva permet dans certains cas de maintenir une connexion sans coupures, en effet elle envoie un message au client ssh après x secondes sans activité (0 = jamais). Si le client répond au serveur, la connexion est maintenue.

```
ClientAliveInterval 600
```

5.11 Autoriser authentification par clé

```
PubkeyAuthentication        yes
```

Laisser yes si vous voulez établir l'authentification par clé comme expliqué plus haut

5.12 Activer les log

```
LogLevel INFO
```

6. Se connecter par la commande SSH

6.1 Authentification par mot de passe

C'est la méthode la plus simple. Depuis la machine cliente, taper :

```
$ssh login@nom du domaine ou adresse IP du serveur
```

- Si c'est la première connexion SSH depuis ce client vers ce serveur, il vous demande si le fingerprint de la clef 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 clef publique et la comparer à celle qu'il vous affiche. Si les deux fingerprints sont identiques, répondez *yes*, et la clef publique du serveur est alors rajoutée au fichier ~/.ssh/known_hosts.
- Si vous vous êtes déjà connecté depuis ce client vers le serveur, sa clef publique est déjà dans le fichier~/.ssh/known_hosts et il ne vous demande donc rien.

Ensuite, entrez votre mot de passe... et vous verrez apparaître le prompt, comme si vous vous êtiez connecté en local sur la machine.

En IPV6 ajouter l'option -6

```
ssh -6 <nom_utilisateur>@<adresse ipv6>
```

6.2 Authentification par clef

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

1. Générer la clefs

La création de la paire de clé se fait avec ssh-keygen.

Il existe 2 types de clés : RSA et DSA. Chacune pouvant être de longueur différente : 1024, 2048, 4096 bits (les clés inférieures à 2048 bits sont à proscrire... surtout les RSA). Pour créer une clé DSA de 2048 bits : ssh-keygen -t dsa -b 2048. Sans paramètres, les options par défaut sont type RSA en 2048 bits.

```
$ssh-keygen –t rsa –b 2048
```

Deux fichiers ont été créés (dans le dossier ~/.ssh/) :

- id_rsa (ou id_dsa dans le cas d'une clé DSA) : contient la clé privée et ne doit pas être dévoilé ou mis à disposition
- id_rsa.pub (ou id_dsa.pub dans le cas d'une clé DSA) : contient la clé publique, c'est elle qui sera mise sur les serveurs dont l'accès est voulu.

L'authentification par clé est active par défaut. Signifie que la valeur par défaut de la clé **PubkeyAuthentication** est **Yes** ; l'authentification par clé est autorisée.

2. *Autoriser votre clef publique*

Pour cela, il suffit de copier votre clef publique dans le fichier ~/.ssh/authorized_keys de la machine sur laquelle vous voulez vous connecter à distance. La commande suivante permet de réaliser cette opération via SSH :

```
$ssh-copy-id –i ~/.ssh/id_rsa.pub   user@ipmachine
```

Ou

```
$scp ~/.ssh/id_rsa.pub    user@ipmachine:/home/user/.ssh/authorized_keys
```

3. *Se connecter*

La commande est la même que pour une authentification par mot de passe mais sans demander le mot de passe

6.3 solution avec ssh-agent

Le serveur SSH est maintenant plus sécurisé, mais taper des passphrases à longueur de journée peut se révéler être très pénible surtout si on a choisi une « vraie » passphrase.

L'agent SSH permet de taper la passphrase une seule fois et de la conserver en mémoire pendant tout son fonctionnement. Les communications SSH fonctionneront donc de façon transparente.

Il faut lancer l'agent avec un shell (le plus simple étant de le lancer avec la variable $SHELL qui contient le shell courant).

Ensuite le programme ssh-add permet de charger les clé présentes dans ~/.ssh/. La passphrase est demandée, toutes les connexions nécessitant les clés chargées par l'agent seront transparentes.

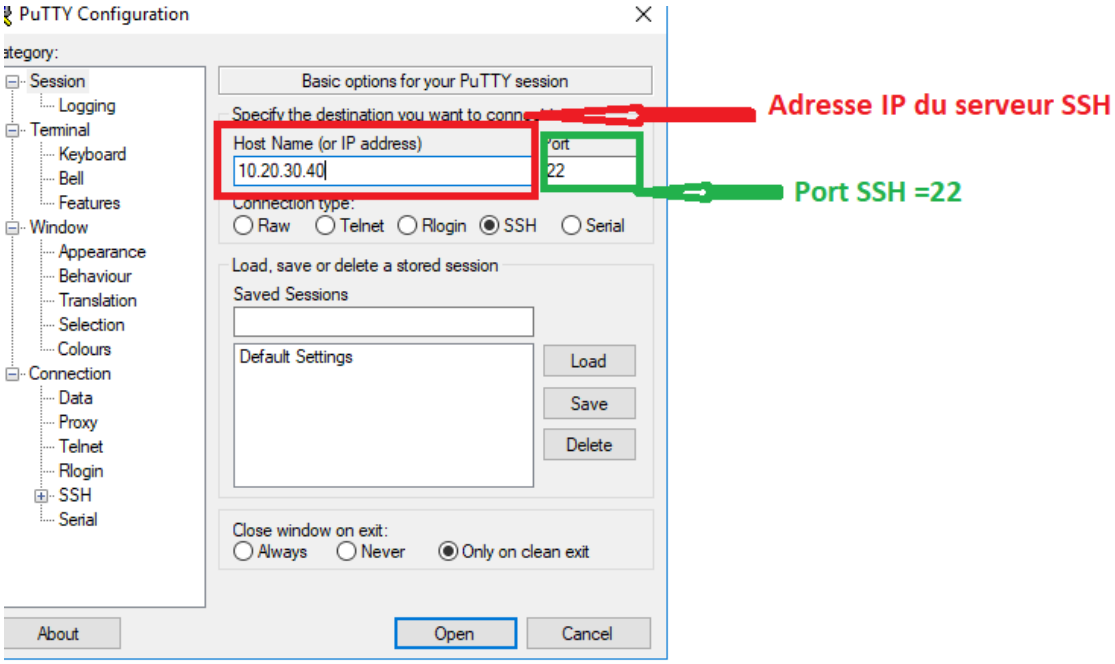
```
$ ssh-agent $SHELL

$ ssh-add
```

7. Se connecter par Putty (client Windows)

Application Putty permet d'accéder au serveur Linux via le protocole SSH depuis une machine Windows

Lancer l'outil et indiquer le nom du domaine ou l'adresse IP du serveur SSH ainsi que le numéro de port 22



8. Transfert

De serveur à serveur depuis votre machine locale

Copie récursive d'un dossier d'un serveur (serveur1) vers un autre serveur (serveur2) depuis votre machine locale.

Cela nécessite d'avoir accès aux deux serveurs, depuis votre machine locale, vous lancez une commande qui copiera les fichiers d'un serveur à un autre.

```
$scp -r -p user@serveur1:chemin/vers/dossier/source user@serveur2:chemin/vers/dossier/destination
```

De serveur à serveur en étant connecté à un serveur

La commande est sensiblement la même, vous êtes connecté sur la machine où sont disponibles les fichiers.

```
$ scp -r -p chemin/vers/dossier/source user@serveur2:chemin/vers/dossier/destination
```

L'option -r indique la récursivité
L'option -p préserve les dates de modification, d'accès, et les modes des anciens fichiers.

En IPV6 ajouter l'option -6

```
scp -6 <élément> <nom>@[addresse ipv6]:<destination>
```

9. Transfert graphique

Nous pouvons utiliser FileZilla comme nous avons vu avec le serveur FTP, mais il faut modifier le port par la valeur 22 ou WinSCP

10. Tunnels et redirection de ports

Voir résumé

11. Réference

https://doc.fedora-fr.org/wiki/SSH_:_Authentification_par_cl%C3%A9
https://fr.wikipedia.org/wiki/Ssh-agent
https://doc.ubuntu-fr.org/ssh
https://buzut.net/empecher-les-timeout-ssh/
https://www.linuxtricks.fr/wiki/ssh-installer-et-configurer-un-serveur-ssh