SSH

Certainement l'un des meilleurs protocoles actuels.

- [SSH] Transfert de disque via SSH
- [SSH] Connexion à distance
- [SSH] Copie de fichier
- [SSH] Serveur OpenSSH

[SSH] Transfert de disque via SSH

Introduction

La manipulation va permettre de transférer le contenu du disque d'une machine A vers une machine B avec SSH à travers le réseau.

Source

- Article d'Adrien Linuxtricks

Manuel

Pour le faire, vous devez démarrer les deux machines sur un live CD car l'opération doit être effectuée à froid pour les disques.

La commande suivante doit être effectuée en root sur la machine du disque dur source :

```
pv /dev/sda | ssh root@<IP_DST> ' cat > /dev/sda'
```

> La commande **pv** permet d'obtenir des informations sur le transfert contrairement à cat qui n'est pas verbeux.

[SSH] Connexion à distance

Introduction

Le **SSH** pour *Secure SHell*, est un protocole dont les fonctionnalités sont multiples et puissantes.

On le connaît essentiellement pour sa prise en main sur des shells distants mais il permet aussi de transférer des fichiers ou faire du port forwarding dans un tunnel chiffré.

Connexion à distance

Manuel

Voici la syntaxe globale pour vous connecter sur une machine distante :

```
ssh |OPTIONS] <USER>@<IP>
```

Options courantes

Options	Descriptions
-i <KEY>	Permet de spécifier une clé privé lors de la connexion.
-p <PORT>	Permet de spécifier un port lors de la connexion.

Génération d'une paire de clés

Bien que la connexion par mot de passe soit fonctionnelle, elle est fortement déconseillée car non sécurisée.

C'est pour cela qu'il est recommandé d'utiliser une paire de clés pour vous authentifier lors de vos sessions SSH.

Cette méthode d'authentification repose sur un **chiffrement asymétrique** avec une **clé publique** et une **clé privée** (vous pouvez retrouver plus d'informations théoriques sur ma documentation).

Voici la commande qui vous permet de générer une paire de clé (fonctionne sur Linux et Windows) :

```
ssh-keygen [ALGO] [BITS_LEN]
```

> Vous pouvez utiliser l'algorithme de chiffrement que vous souhaitez mais je préfère utiliser celui par défaut qui est léger et très sécurisé. Je n'utilise donc pas d'option à cette commande.

Il vous sera ensuite demandé le chemin de destination de la nouvelle clé (par défaut dans le répertoire **~/.ssh**).

Vous pouvez aussi définir une passphrase ce qui permet une authentification à double facteur (clé+mot de passe), mais cette option est facultative.

Copie de la clé publique sur le serveur

Pour que l'authentification par clé fonctionne, il vous faut copier la clé publique dans le fichier **~/.ssh/authorized_keys** sur le serveur distant.

Vous pouvez le faire manuellement ou automatiquement grâce à la commande ssh-copy-id :

```
ssh-copy-id -i <PUBLIC_KEY> <USER>@<IP>
```

[SSH] Copie de fichier

Introduction

Le protocole SSH prends en charge le transfert de fichiers via **SCP** ou **SFTP**.

SCP

Bien que cette commande soit dépréciée, elle est simple d'utilisation et fonctionne parfaitement pour transférer un fichier rapidement d'une machine à l'autre. Le tout dans un tunnel SSH qui est sécurisé car chiffré.

Syntaxe globale

```
scp [OPTION] <SOURCE> <DESTINATION>
```

Téléchargement d'un fichier

```
scp <USER>@<IP>:<REMOTE_FILE_PATH> .
```

> Le caractère **.** symbolise que le fichier va être copié dans le répertoire courant mais il est tout à fait possible de spécifier un chemin à la place et même un nouveau nom (comme pour la commande cp).

Envoi de fichier

```
scp <FILE_TO_SEND> <USER>@<IP>:<DESTINATION_FILE_PATH>
```

SFTP

Le protocole **SFTP** est équivalent au protocole FTP mais fonctionnant dans un tunnel SSH ce qui chiffre la connexion.

> Le protocole SFTP n'est pas le même protocole que FTPS qui est l'équivalent du FTP + SSL.

Syntaxe globale

La syntaxe est similaire à la commande ssh qui permet de se connecter à un hôte distant :

```
sftp <USER>@<IP>
```

Quelques commandes

Une fois dans le shell sftp, les commandes disponibles sont les mêmes que la commandes FTP :

Options	Descriptions
put <FILE>	Permet d'envoyer un fichier sur le serveur distant.
get <FILE>	Permet de récupérer un fichier depuis le serveur distant.

[SSH] Serveur OpenSSH

Introduction

La bonne configuration d'un **serveur SSH** est primordiale pour sécuriser votre serveur des attaques extérieures.

C'est pourquoi je vous recommande de consulter le guide de l'ANSSI qui décrit les bonnes pratiques à avoir.

Nous verrons dans ce tutoriel, uniquement des configurations classiques et non-exhaustives.

Source

- Guide de l'ANSSI OpenSSH

Installation

Installez le paquet **openssh-server** sur votre serveur (le nom peut varier selon les distributions) :

```
sudo apt install -y openssh-server
```

Configuration

Fichier de configuration

Le fichier de configuration du serveur SSH est **/etc/ssh/sshd_config** :

```
nano /etc/ssh/sshd_config
```

>

> Vous pouvez activer ou désactiver des options en choisissant de les commenter ou non (caractère**#**).

Changer le port d'écoute

Option intéressante pour éviter d'être dans le viseur des bots qui brute force les serveurs SSH exposés sur Internet .

Par exemple, vous pouvez remplacer le port **22** définit par défaut par le port 2222 :

```
Port 2222
```

Changer l'interface d'écoute

Par sécurité, il peut être intéressant de définir l'interface d'écoute du serveur SSH. Par défaut, toutes les interfaces sont en écoute.

Prenons l'exemple d'un serveur appartenant à deux réseaux :

Nom du réseau	Adresse IP
LAN	192.168.1.10
DMZ	10.0.0.10

Vous pouvez donc décider d'activer l'administration par SSH **seulement via le LAN** pour protéger votre serveur grâce à l'option suivante :

```
ListenAddress 192.168.1.10
```

Connexion root

Cette option n'est pas recommandée mais elle est importante à connaître.

Elle permet d'activer ou non la connexion en root et le type d'authentification autorisé.

- Si vous souhaitez **interdire la connexion root** :

```
PermitRootLogin no
```

- Si vous souhaitez **autoriser la connexion root par mot de passe et par clé** :

```
PermitRootLogin yes
```

- Si vous souhaitez **autoriser la connexion root uniquement par clé** :

```
PermitRootLogin prohibit-password
```