SSH : Administration distante sécurisée

SSH (secured shell) est un protocole et le service permettant une **connexion distante** vers un équipement en vue de son **administration système** (gestion des services, installations, modification du paramétrage matériel, etc).

Caractéristiques

- Protocole sécurisé : la **connexion est chiffrée** par une technologie asymétrique (RSA, DSA, ...)
- Port TCP/22
- Possibilités d'authentification
  - par comptes utilisateurs de l'environnement (PAM)
  - par clé privée/certificat pour le client

Pour des **raisons de sécurité**, l'établissement d'une **connexion** en SSH avec un **compte super-utilisateur n'est plus autorisée** (il existe des commandes pour lever cette interdiction, au risque d'ouvrir une faille de sécurité dans le système ; voir ci-après).

Il est donc nécessaire de suivre la procédure suivante :

1. créer un compte non *super-utilisateur* sur la machine hébergeant le service SSH
2. se connecter avec le *client SSH* de votre choix, en utilisant ce compte
3. une fois la connexion établie (et les échanges sécurisés), basculer en *mode super-utilisateur*

Contributeurs

(SISR2-2016) Anthony Varin, Alexandre Saligny

Mise en route et utilisation

Installation du paquet ssh

Utiliser la commande :

```
 apt install ssh
//ou
apt install openssh-server
```

Connexion distante

La connexion distante à une machine hébergeant le service SSH s'établit par la commande ***ssh*** ou un client dédié comme ***Putty***.

```
ssh -l <nomCompte> <IPServeurDistant>
//ou
```

```
ssh <nomCompte>@<IPServeurDistant>
```

Sécurisation

Accès root

Pour définir l'accès distant avec un compte *root*, on éditera le fichier ***/etc/ssh/sshd_config*** , pour adapter la ligne :

```
PermitRootLogin {yes|no}
```

Suite au paramétrage, on redémarre le service :

```
systemctl restart ssh
```

L'accès distant en root est interdit par défaut.

On n'autorisera que pour des raisons très spécifiques en mesurant les risques qu'on encoure.

Limitation de la navigation

Par défaut, un utilisateur autorisé à se connecter en SSH est positionné dans son dossier ***/home***, mais il peut **naviguer sur l'intégralité de la machine** distante.

Cela pose de sérieux **problèmes de sécurité** :

- **accès illicite** à des contenus privés,
- **destruction de données**,
- installation de **logiciels malveillants**,
- modification ou **altération de contenus** (pages d'un site, paramètres d'un service, mots de passe, données d'une base, etc).

Blocage dans un dossier (chroot)

Dans le fichier de configuration de SSH (**/etc/ssh/sshd_config**), on peut ajouter des paramétrages pour un utilisateur (**Match User <nomUser>**) ou pour un groupe (**Match Group <nomgroupe>**).

Notamment, on **limitera** l'utilisateur :

1. à l'**accès à un dossier** avec le **paramètre ChrootDirectory**
2. à une **fonction précise** (échange de données en SFTP ou **scp**) avec le **paramètre ForceCommand**

Exemple pour un **groupe webDepot** dédié aux développeurs autorisés à **déposer des fichiers** dans le **dossier de Apache** :

```
Match Group webDepot
        ChrootDirectory /var/www/html/
        ForceCommand internal-sftp
        AllowTcpForwarding no
```

Remarque Le dossier attribué à ChrootDirectory doit être en écriture uniquement pour le

```
GatewayPorts no
X11Forwarding no
```

compte **root**

Détection et blocage du brute force

SSH est une porte d'entrée très prisée des hackers : une fois entrés sur la machine, il ne leur reste plus qu'à exploiter les failles locales.

On peut chercher à **limiter** leurs **tentatives d'intrusion** par **brute force** en utilisant Fail2ban pour SSH (ou en interne).

Fail2ban propose 4 modes de blocage :

- **normal** : s'appuie sur les connexions d'utilisateur en échec (mauvais compte/mot de passe, mauvais certificat, etc)
- **extra** : ajoute les erreurs de négociations (de version, de protocole de chiffrement), méthode qui peut être utilisée pour détecter des failles en listant toutes les possibilités
- **ddos** : ajoute la prise en compte des demandes de connexions non abouties (les machines pirates lancent des demandes de connexion puis les interrompent pour saturer le service)
- **aggressive** : normal + ddos + extra

Connexion par clé privée / certificat

L'**authentification par compte / mot de passe** devient de plus en plus une **fragilité des systèmes**, avec des risques multiples :

- mots de passe faibles
- phishing
- attaques par dictionnaire ou brut force
- données volées

Pour SSH en particulier, le risque est d'autant plus important qu'il s'agit de permettre l'accès à une machine pour y opérer des actions sensibles : alimenter le contenu d'un site, paramétrer l'ordinateur, etc.

Une solution **plus robuste** consiste à garantir la connexion par un **système fort de clé privée/clé publique**.

1. l'utilisateur génère sa clé privée
2. on génère ensuite sa clé publique (soit lui-même, soit en passant par une autorité)
3. on fourni la clé publique de l'utilisateur à la machine à laquelle on accèdera en SSH
4. on paramètre la connexion (par exemple avec Putty) pour utiliser la clé privée du client lors de la connexion

Une démarche complète et bien expliquée est présentée ici :

https://www.tutos-informatique.com/comment-activer-lauthentification-par-cle-ssh-sur-votre-serveur/.

Gérer l'établissement d'une connexion distante récurrente

Lorsque l'on souhaite établir une connexion distante récurrente sans nécessiter de renseigner le compte et le mot de passe à chaque connexion (par exemple pour automatiser une sauvegarde), on réalisera la procédure suivante :

- créer une clé et son certificat (clé publique .pub) sur une machine qui devra accéder à une cible de façon récurrente
- copier le certificat (clé publique) sur la cible

Démarche

Création des éléments de sécurité

Pour créer la clé et son certificat, on utilisera la commande :

```
ssh-keygen –t {rsa|dsa} [-b <nombre_bits>]
```

- <nombre_bits> est une puissance de 2 précisant la complexité de la clé (1024, 2048, etc)

une fois la commande exécutée ce type de message apparaît.
On peut renseigner la *passphrase* si on veut sécuriser la connexion par un code d'accès

```
 Generating public/private rsa key pair.
    Enter file in which to save the key (/~root/.ssh/id_rsa):
    Enter passphrase (empty for no passphrase):
    #(la passphrase crypte la clé publique est augmentent ainsi la sécurité
de la communication)
    Enter same passphrase again:
    Your identification has been saved in /~root/.ssh/id_rsa.
    Your public key has been saved in /~root/.ssh/id_rsa.pub.
    The key fingerprint is:
    da:21:b7:c9:ef:81:76:de:80:02:54:51:06:03:17:25
    The key's randomart image is:

    +--[ RSA 2048]----+
    |    ..E*+        |
    |     o +         |
    |    .            |
    |   .             |
    |    . . S        |
    |     . * *       |
    |      o O +      |
    |       o + +     |
    |         .+ .    |
```

```
    +-----------------+
```

Copie de la clé sur la cible

On peut alors recopier la clé sur la cible, ce qui permettra ensuite d'établir une connexion SSH sans interaction avec le <compte> utilisé pour la copie.

```
ssh-copy-id -i <chemin/clé_publique> <compte>@<IP_ou_FQDN_cible>
```

- la clé publique générée par *ssh-keygen* se trouve par défaut dans ***/root/.ssh/*** sous le nom ***id_rsa.pub***.

A TRIER et replacer au bon endroit La commande **ssh-add** : permet de sauvegarder la ***passphrase*** dans une mémoire cache, car sinon il redemande à chaque fois qu'on relance le script, la passphrase. Mais si on éteint la machine, il oublie la passphrase.

Si la commande ne fonctionne pas il faut utiliser la commande **ssh-agent /bin/bash** puis retaper la commande **ssh-add**

Installation du paquet **sshpass** avec la commande :

```
apt install sshpass
```

Pour éviter de devoir renseigner le mot de passe à chaque fois dans le script .

Script.bash :

```
#! /bin/bash
echo off
while read nom ip
  do
    sshpass –f motpasse.txt ssh-copy-id –i id_rsa.pub $nom@$ip
    echo –e "machine" $ip "déployée" >> sshpass.log
done < ./machine.csv
```

- **#! /bin/bash** (pour afficher la syntaxe en couleur)
- **echo off** (pour ne pas afficher l'echo présent plus bas)
- **while read nom ip** (lit chaque donnée du fichier)
- **eo** (pour faire une boucle)
- **sshpass –f motpasse.txt ssh-copy-id –i id_rsa.pub $nom@$ip** (sshpass –f motpasse.txt sert à dire que à chaque demande du mot de passe l'information se trouve dans le fichier texte, ssh-copy-id –i id_rsa.pub pour envoyer la clé publique au machine définie dans le fichier csv)
- **echo –e "machine" $ip "déployée" » sshpass.log** (cette ligne sert à créer un fichier log qui dit quelle adresse Ip est déployée)
- **done<./machine.csv** (done pour dire que la boucle à exécuter est terminée, et on précise le fichier dont on lit le contenu)

Installer le paquet ssh sur les postes utilisateurs avec la commande : **apt-get install**

ssh