Configurer une authentification avec un couple de clés privée/publique SSH

Présentation

Pour administrer un serveur Linux, vous pouvez utiliser le compte **root** ou, ce qui est fortement conseillé, un compte que vous avez créé et à qui vous avez permis une **élévation de privilèges** pour permettre l'utilisation de **sudo**.

Si vous gérez un autre serveur, il est également fortement conseillé d'utiliser **un mot de passe différent**. Cette solution n'est **pas satisfaisante et peu sécurisée** si vous devez gérer de nombreux serveurs.
Par ailleurs cela est problématique si vous avez des tâches d'administration à **automatiser** car la saisie manuelle du mot de passe sera nécessaire ou bien il faudra indiquer le **mot de passe dans les scripts** ce qui est problématique si vous n'avez pas de solution pour les chiffrer. Les solutions possibles :

- utiliser un **annuaire LDAP pour centraliser** la gestion des comptes.
- utiliser des **clés SSH publique**.

Vous aller configurer le compte linux que vous avez créé afin de permettre d'ouvrir une session en utilisant une **clé publique SSH** :

- Vous utiliserez **votre propre clé publique SSH** pour vous connecter.
- Vous permettrez à l'enseignant de se connecter en simple utilisateur avec un compte que vous devez créer et appeler ensbtssio avec sa **clé publique SSH**.

Après la création de votre **couple de clés Privée/publique**, communiquez aux enseignants votre **clé publique** dans le dossier partagé de l'équipe Teams.

> - En utilisant **mot de passe**, vous utilisez **un seul facteur** d'authentification.
> - En utilisant une **clé publique SSH,** vous utilisez également **un seul facteur** d'authentification. Ce facteur utilise une solution de chiffrement et est alors considéré comme une **authentification forte** (ANSSI).

Vous pouvez utiliser la **même clé publique SSH sur plusieurs serveurs** en ne retenant qu'un seul mot de passe, celui de la passphrase qui protège de votre clé privée.

> Après avoir configuré une authentification avec une clé SSH publique, il est conseillé de **désactiver** l'authentification par mot de passe sur les serveurs.

Pour en savoir plus : https://www.it-connect.fr/chapitres/authentification-ssh-par-cles/

La confiance entre client et serveur

Les clefs SSH permettent aussi de garantir l'identité d'un interlocuteur sur le réseau sans avoir à échanger de secret partagé :

- Le client qui se connecte à un serveur par SSH doit avoir **confiance en l'identité de ce serveur** (éviter les attaques de type attaque de l'**homme du milieu**).
- Pour accorder l'accès de client au système, le serveur doit avoir **confiance en l'identité du client**.

Authentification du client par le serveur

Pour que le serveur fasse confiance au client qui tente de se connecter, l'administrateur du système doit placer la **clef SSH publique de chaque utilisateur·rice** dans le fichier **~/.ssh/authorized_keys** du compte.

Lors de la connexion au serveur, le client utilise la clef SSH privée de l'utilisateur·rice et le serveur peut vérifier l'authenticité grâce à la clef publique correspondante conservée dans le fichier **~/.ssh/authorized_keys** du compte auquel le client tente de se connecter.

Authentification du serveur par le client

Lors de sa première connexion à un serveur, le client enregistre une clef publique fournie par le serveur en supposant qu'il s'agit bien de la clef serveur de l'hôte auquel il cherche à se connecter.

Avant de faire confiance au serveur, le client va demander à l'utilisateur·rice d'accepter la connexion et l'enregistrement de la clef du serveur.

Cela se présente de cette manière :

```
~$ ssh lab.example.org
The authenticity of host 'lab.example.org (192.168.1.3)' can't be established.
ECDSA key fingerprint is SHA256:wxocsOPNEfYuuTcXGuitKR8RUowpGbgpXTEsXXirAFI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'lab.example.org' (ECDSA) to the list of known hosts.
```

Si l'utilisateur·rice accepte de se connecter à ce serveur, le client va enregistrer la clef du serveur dans son fichier **~/.ssh/known_hosts** (indexée avec une valeur de hashage calculée avec le nom du serveur tel qu'il a été donné au client SSH).

La commande **ssh-keygen -F <hôte>** permet de rechercher la clef d'une machine dans le fichier **~/.ssh/known_hosts**.

```
~$ ssh-keygen -F lab.example.org
# Host lab.example.org found: line 133
|1|4M/wUtH5ptcDaJ7BwKPcpci6iZA=|Ba7Gk24pQ4qdIg4hR5KVIjdGo7Y= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNo(...)3+Ciz9DgKW2yAX7g26E73LWSI=
```

Lors des connexions suivantes, le client va rechercher de la même façon dans le fichier **~/.ssh/known_hosts** la clef publique du serveur :

- Si la clef publique ainsi conservée correspond bien à la clef privée utilisée par le serveur, le client va faire confiance au serveur et procéder ensuite à sa propre authentification.
- Si ce n'est pas le cas, le client interrompt la connexion et affiche à l'utilisateur·rice un message lui indiquant que le serveur auquel il ou elle tente de se connecter n'a plus la même clef SSH que lors de sa dernière connexion, ce qui indique que le serveur n'est peut-être pas celui auquel il ou elle souhaite se connecter.

On remarque qu'une **faiblesse du protocole** existe à ce niveau puisqu'à la première connexion, l'utilisateur·rice qui accepte aveuglément la clef SSH du serveur prend le risque de se connecter à un serveur qui n'est pas celui qu'il croit être.

Ce risque peut être écarté si les clefs publiques des serveurs sont distribuées par d'autres moyens aux utilisateur·rice·s pour les ajouter à leur fichier known_hosts, leur permettant ainsi d'authentifier le serveur dès leur première connexion.

Une variante de cette solution est facilitée dans les versions récentes d'OpenSSH où, comme on le voit dans l'exemple ci-dessus, au lieu de répondre yes ou no à la question du client demandant s'il faut se connecter à ce serveur, celui-ci donne la possibilité de fournir l'empreinte de la clef SSH attendue. La connexion se fera alors uniquement si celle-ci correspond bien à l'empreinte de la clef donnée par le serveur (ce qui est plus fiable que de vérifier visuellement l'empreinte affichée par le client).

Générer une paire de clé SSH depuis un client OpenSSH

Générer une paire de clés privée/publique depuis un client Windows ou linux.

> Il est conseillé de protéger l'utilisation de la clé privée avec une passphrase.

```
$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/centrecallbd/.ssh/id\_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/centrecallbd/.ssh/id\_ed25519.
Your public key has been saved in /home/centrecallbd/.ssh/id\_ed25519.pub.
The key fingerprint is:
SHA256:0rjedyVuT2fzEJHgw5I9lfmTsQ6MHSD87Xrr/aXE3r4 centrecallbd@Ch2Lab1
The key's randomart image is:
+---[RSA 2048]----+
|        .. .o .o |
|         ..= +oo |
|          + @ +.+|
|       o   + B =.|
|      o S   . + .|
|       o    .o.o |
|      .    ..o=.+|
|     . .  ..+=.*+|
|      . .. oo+=EB|
```

```
+----[SHA256]-----+
$
```

Dans le dossier caché **.ssh** (sous Windows Linux ou MacOsX) vous avez votre couple de clés privée (id_ed25519) et publique (id_ed25519.pub).

```
$ ls .ssh
id_ed25519  id_ed25519.pub
$
```

Un autre fichier **know_hosts** sera ensuite créé dans le dossier **.ssh** afin de contenir **les clés publiques des serveurs** sur lesquels vous vous êtes authentifié avec un mot de passe ou une clé SSH publique.

- Pour retrouver l'entrée d'un nom d'hôte connu dans known_hosts:

```
$ ssh-keygen -H  -F <hostname or IP address>
```

- Pour supprimer une seule entrée de known_hosts:

```
# ssh-keygen -R <hostname or IP address>
```

Configurer un accès SSH avec une clé SSH depuis un client OpenSSH

- Copiez ensuite votre clé publique sur le serveur auquel vous souhaitez accéder avec la clé SSH.

```
$ ssh-copy-id utilisateur@IP_ordinateur_cible
```

La clé publique est copiée dans le fichier **.ssh/authorized_keys** du serveur distant.

La commande ssh-copy-id n'est pas disponible sous Windows. Vous pouvez alors :

- utiliser la commande **scp** pour copier le fichier id_ed25519.pub dans le dossier l'utilisateur ;
- ouvrir une session ssh pour pouvoir ensuite ajouter le contenu du fichier id_ed25519.pub dans le fichier authorized_keys :

```
C:> scp .ssh/id\_ed25519.pub compteutilisateur@adresseip:/home/compteutilisateur/
C:> ssh compteutilisateur@adresseip
$ cat id\_ed25519.pub >> .ssh/authorized_keys
```

Vous devez maintenant pouvoir vous connecter sans mot de passe au serveur distant :

```
$ ssh utilisateur@IP_ordinateur_cible
```

Il est fortement conseillé ensuite de désactiver l'authentification par mot de passe en modifiant le fichier de configuration du service ssh sur le serveur distant **/etc/ssh/sshd_config** :

- Décommentez la ligne suivante en mettant sa valeur à **no** :

```
PasswordAuthentication no
```

- Modifiez la ligne suivante pour mettre sa valeur à **no** :

```
ChallengeResponseAuthentication no
```

```
* sauvegardez le fichier **/etc/ssh/sshd\_config** et relancer le service ssh :
```

```
$ sudo systemctl restart ssh
```

Accès au serveur en SSH avec Putty

- Configurez Putty pour avoir un accès **console** à votre serveur.

- Indiquez l'**adresse IP** de votre serveur et le **port 22 (SSH)**.
- précisez le compte **root** pour vous connecter :
- précisez votre **clé privée**