Authentification SSH par clés

Jusqu'à présent, nous avons pris pour habitude d'utiliser comme seul facteur d'authentification le mot de passe. **Il faut néanmoins savoir qu'il est également possible d'utiliser un autre facteur, une clé**. Cela est parfois préféré car la clé permet de ne pas avoir à retenir systématiquement des mots passe différents. Nous allons ici voir le fonctionnement général de cette méthode d'authentification.

Il est courant sur des serveurs SSH de n'autoriser uniquement l'authentification par clé afin de sécuriser ce protocole. La plupart du temps la génération de la clé se fait sous Linux sans trop de problème étant donné qu'*OpenSSH* y est nativement présent. Sous *Windows* toutefois, quelques manipulations sont à effectuer afin de générer et d'envoyer notre clé au serveur.

L'authentification par clés se fait donc via une paire de clés, **le client va donc générer une paire de clés, une publique et une privée**. Il va bien entendu garder sa clé privée pour lui et envoyer la clé publique au serveur SSH qui la stockera dans un endroit prévu cet effet.

Si vous souhaitez avec plus d'informations sur le système de clé privée/clé publique, je vous invite à lire ce cours écrit par Florian Burnel :

- Clés asymétriques :

I. Génération de la clé

Étant donné qu'un client SSH peut être un client *Linux* ou un client *Windows*, nous allons voir comment générer cette paire de clés sous Linux en ligne de commande, et sous *Windows* via *PuttyGen*.

Générer une paire de clés sous Linux

Voyons tout d'abord comment générer une paire de clés sous Linux en ligne de commande. Pour cela, nous allons utiliser la commande "*ssh-keygen*" :

```
ssh-keygen
```

Si aucune option n'est spécifiée, une clé *RSA* de 2048 bits sera créée, ce qui est acceptable aujourd'hui en termes de sécurité. Si vous souhaitez spécifier une autre taille de clé, vous pouvez utiliser l'option "*-b*" :

```
ssh-keygen -b 4096
```

Par défaut, la clé va être stockée dans le répertoire *.ssh/* de l'utilisateur courant (Exemple : */root/.ssh* pour l'utilisateur *root*).

> **Note** : Pour une sécurité correcte de vos communications, il est aujourd'hui recommandé d'utiliser des clés de 4096 bits.

Il va ensuite nous être proposé de saisir une *passphrase*, **je vous recommande d'en mettre une** ! Concrètement, nous avons vu qu'une clé pourra être envoyé à plusieurs serveurs pour éviter d'avoir à saisir un mot de passe, en tant que possesseur de la clé privée correspondant à la clé publique envoyée au serveur SSH sur lequel on souhaite se connecter, le serveur nous acceptera directement.

Néanmoins, **si un tiers parvient à nous dérober notre clé privée, il arrivera à se connecter aux serveurs sans mot de passe**. Ainsi, une passphrase permet la protection de notre clé privée via un mot de passe, ou plutôt une phrase de passe ("*passphrase*"). L'avantage par rapport à un mot de passe SSH est que vous n'avez qu'un mot de passe à retenir, celui de votre clé privée et pas un mot de passe par serveur SSH.

Une fois créées, vous pourrez donc voir vos clés dans le répertoire "*.ssh*" de l'utilisateur :

```
root@itc-serveur-01:~# ls -al .ssh

total 20

drwx------ 2 root root 4096 juin 8 11:15 .

drwx------ 10 root root 4096 juin 8 11:11 ..

-rw------- 1 root root 3247 juin 8 11:18 id_rsa

-rw-r--r-- 1 root root 745 juin 8 11:18 id_rsa.pub

-rw-r--r-- 1 root root 444 avril 23 03:34 known_hosts
```

Pour rappel, nous nous situons toujours ici sur un Linux client, on remarque l'existence d'un autre fichier "*known_hosts*", il s'agit ici d'un fichier permettant d'identifier un serveur. Si vous vous connectez en SSH à plusieurs serveurs depuis votre utilisateur ("*root*" dans mon cas), votre fichier *known_host* va peu à peu se remplir. Cela entraîne entre autre le fait qu'une demande validation de la clé serveur est demandée à la première connexion du serveur mais pas lors des connexions suivantes.

II. Mettre la clé sur le serveur

Il nous faut maintenant faire l'opération consistant à envoyer notre clé publique sur le serveur afin de pouvoir s'y authentifier, sous *Linux*, une commande est prévue à cet effet. ("*ssh-copy-id*" sous *Linux*). Il nous faut pour cela établir une connexion *SSH* sur le serveur par mot de passe (une dernière fois) avec la commande "*ssh-copy-id*" :

```
ssh-copy-id utilisateur@serveur
```

Par exemple si je souhaite, en tant que client, ajouter ma clé sur le serveur *SSH 192.168.240.132* pour l'utilisateur *root* :

```
ssh-copy-id root@192.168.240.132
```

Notez que par défaut, cette commande va envoyer la clé publique "*.ssh/id_rsa.pub*". On peut néanmoins, si l'on possède plusieurs paires de clés, spécifier la clé publique à envoyer au serveur, par exemple :

```
ssh-copy-id -i ~/.ssh/id_rsa_groupeServeurA.pub root@192.168.240.132
```

On devra donc saisir une dernière fois le mot de passe *SSH* de l'utilisateur visé sur le serveur *SSH*, puis nous aurons quelques messages de validation :

```
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
```

```
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
```

```
Number of key(s) added: 1
```

Ici, on voit donc qu'une nouvelle clé a été ajoutée au serveur. Le message qui suit nous le valide :

```
Now try logging into the machine, with: "ssh 'root@192.168.240.132"
```

```
and check to make sure that only the key(s) you wanted were added.
```

En effet, maintenant que notre clé publique est présente sur le serveur que nous possédons dans notre répertoire *.ssh* sa clé privée associée, nous allons pouvoir nous connecter au serveur juste en saisissant la commande suivante :

```
ssh utilisateur@serveur
```

La passphrase de la clé publique nous sera demandée, puis on sera connecté en SSH à notre serveur.

L'avantage est le suivant : si cette même clé publique est envoyée à 30 autres serveurs, **la passphrase sera la même pour toutes les connexions alors qu'il faudra retenir 30 mots de passe différents** si vous utilisez la méthode d'authentification habituelle par mot de passe.

Dans le cas où vous êtes sous *Windows* ou alors que la méthode précédemment proposée n'est pas envisageable, on peut aussi faire cette manipulation manuellement. Il nous suffira alors simplement de copier le contenu de notre clé publique (par défaut "*.ssh/id_rsa.pub*") dans le fichier "*.ssh/authorized_keys*" de l'utilisateur du serveur visé.

Nous supposons ici que nous voulons une connexion sur le compte "*root*" (c'est bien sûr déconseillé dans un cadre réel). Nous allons donc aller dans le dossier "*home*" de cet utilisateur et créer le dossier "*.ssh*" s'il n'existe pas déjà :

```
mkdir .ssh
```

Le dossier "*.ssh*" est là où vont se situer les informations relatives aux comptes et aux connexions SSH de l'utilisateur. Nous y créerons le fichier "*authorized_keys*" s'il n'existe pas. Ce fichier contient toutes les clés publiques que permettent d'établir des connexions avec le serveur et l'utilisateur dans lequel il se trouve (selon le dossier "*home*" dans lequel il se trouve). Nous allons ensuite dans ce fichier mettre notre clé publique (éviter de mettre les "*=== BEGIN===*" et "*===END===*" générés automatiquement par *PuttyGen*).

> **Note** : Sous *Windows*, il est possible de récupérer le contenu de la clé publique en l'ayant ouverte avec un simple bloc-notes pour le transférer dans le fichier "*authorized_keys*".

Il nous faudra ensuite donner les droits les plus restreints à ce fichier par sécurité et pour les exigences d’OpenSSH :

```
chmod 700 ~/.ssh -Rf
```

Notre clé est maintenant présente sur le serveur, il nous reste plus qu’à nous connecter pour tester !

Pour les curieux, vous pourrez aller voir sur votre serveur SSH le contenu du fichier "*authorized*_keys" qui se situe dans le répertoire *.ssh* de l'utilisateur destinataire. Oui, notez bien qu'une connexion SSH est une autorisation faite sur un utilisateur précis de la machine serveur. On peut avoir les identifiants pour se connecter à une machine Linux en tant qu'utilisateur "*mickael*" par exemple, mais pas en tant que root, ou inversement. Il s'agit en fait de l'accès à un compte sur une machine plutôt qu'à une machine, simple précision technique. ?

Révision #3
Créé 2020-09-23 21:57:15 UTC par garfieldtux
Mis à jour 2023-03-04 18:59:23 UTC par garfieldtux