SSH c’est quoi ?

Avant de voir comment contrôler la Raspberry Pi avec SSH, voyons exactement ce qu’est SSH.

SSH (pour Secure SHell) désigne à la fois un logiciel et un protocole de communication informatiques. Ce protocole possède par ailleurs la particularité d’être entièrement chiffré. Cela signifie donc que toutes les commandes que vous exécuterez via SSH seront totalement secrètes !

SSH a été créé en 1995 avec pour principal but de permettre la prise de contrôle à distance d’une machine à travers une interface en lignes de commande.

Aujourd’hui, SSH est principalement utilisé à travers l’implémentation libre OpenSSH qui est présente dans la plupart des distributions Linux. Comment fonctionne SSH ?

Nous ne rentrerons pas ici dans les détails cryptographiques et autres. Nous nous contenterons de faire un rapide tour d’horizon afin que vous saisissiez un peu mieux comment utiliser SSH.

De façon générale, SSH permet de se connecter à distance sur une machine en utilisant un compte utilisateur de la dite machine.

Pour cela, l’ordinateur qui doit se connecter à la machine distante va fournir à celle-ci le nom de l’utilisateur à utiliser et son mot de passe. Il est possible dans certains cas d’utiliser un jeu de certificats sur l’ordinateur et la machine distante, permettant ainsi d’obtenir une connexion sécurisée sans avoir à taper un mot de passe. Il s’agit là d’un cas d’utilisation plus poussé que nous n’aborderons pas ici.

Par défaut SSH propose uniquement une prise de contrôle en lignes de commande. Il est possible dans certains cas d’ajouter une interface graphique mais il s’agit là d’une méthode plus complexe que nous ne verrons pas ici. Installer SSH pour prendre le contrôle de votre Raspberry Pi

Maintenant que nous en savons un peu plus sur SSH, voyons comment le mettre en place pour contrôler votre Raspberry Pi !

Tout d’abord, vous devez savoir que l’installation de SSH se découpe en deux parties. En effet, vous aurez besoin d’un côté d’un serveur SSH sur votre Raspberry Pi et d’un autre d’un client SSH sur votre ordinateur. Le premier recevra les commandes à lancer tandis que le second les enverra. Activer le serveur SSH sur votre Raspberry Pi

Pour un tutoriel un peu plus complet, voir comment activer SSH sur le Raspberry Pi.

Par défaut, SSH est installé sur la Raspberry Pi, mais est désactivé pour des raisons de sécurité. La première chose à faire sera donc d’activer SSH sur votre Raspberry Pi.

Pour cela, il vous suffit de brancher la carte MicroSD de votre Raspberry Pi sur votre ordinateur, de vous rendre sur la carte, et de créer un fichier nommé ssh dans la partition boot. Nous parlons bien ici de la partition boot (celle montée par défaut sous Windows), et non du dossier /boot/ présent dans la partition principale de la carte SD.

Une fois ceci fait, il ne vous reste qu’à remettre votre carte SD dans votre Raspberry Pi, et SSH sera activé à son prochain démarrage.

Notez bien que maintenant que SSH est activé, tout compte créé sur votre Raspberry Pi peut être utilisé pour se connecter via SSH. Assurez-vous donc d'employer des mots de passe forts (et notamment de changer le mot de passe par défaut de l'utilisateur pi)! Installer le client SSH sur votre ordinateur

Cette fois, pas de bonne nouvelle, à priori vous n'avez pas de client SSH installé chez vous. Il y a donc deux possibilités, soit vous avez un ordinateur sous Windows (et nous nous demandons bien pourquoi :b), soit vous avez un ordinateur sous Linux (et les femmes – ou les hommes d'ailleurs – se jettent à vos pieds, vos ennemis vous craignent et vos amis vous envient).

Nous allons donc voir l'installation d'un client SSH pour les deux cas. Installer un client SSH sur Linux

Si vous êtes sous Linux, vous le savez déjà, le monde est plus beau ! Pour installer un client SSH, rien de plus simple, il vous suffit de lancer la commande suivante :

sudo apt-get update && sudo apt-get install openssh-client

Vous voyez, tout est plus simple sous Linux (oui, nous sommes de mauvaise foi) ! Installer un client SSH pour Windows

Vous avez fait le mauvais choix mais il vous sera beaucoup pardonner (enfin peut-être)…

Vous allez devoir installer le logiciel Putty qui est un client SSH et TelNet. Vous pourrez trouver cet excellent outil sur le site dédié.

Point intéressant, Putty fait partie de ces logiciels qui ne nécessitent pas d'installation ! Utiliser SSH pour vous connecter à la Raspberry Pi

Maintenant que nous avons installé un serveur et un client SSH, il ne nous reste plus qu'à les utiliser.

Là encore, deux méthodes différentes selon que vous soyez sous Linux (béni soit le grand pingouin) ou sous Windows (que se brise ses carreaux).

Dans le fond, le principe reste néanmoins le même, utilisez le client SSH pour communiquer avec le serveur, ceci à l'aide de l'adresse du serveur, d'un nom utilisateur et du mot de passe associé. Utilisez SSH avec Linux (ou Mac OS X)

Comme toujours avec Linux, il vous suffit d'une seule ligne de commande pour vous connecter à la Raspberry Pi.

ssh utilisateur@adresse_ip_ou_url_serveur

Bien entendu, vous devrez remplacer « utilisateur » par le nom de l'utilisateur avec lequel vous souhaitez vous connecter et « adresse_ip_ou_url_serveur » par l'IP du serveur ou son adresse URL si il en possède une.

Une fois que vous aurez rentré le mot de passe du compte utilisateur (lequel, pour des questions de sécurité, ne s'affiche pas quand vous le tapez), vous serez connecté au terminal de la Raspberry Pi et toutes les commandes tapées seront faites sur la Raspberry Pi ! Pour quitter SSH il vous suffit d'utiliser la commande « exit ».

Si vous utilisez un Mac, c'est la même chose, ouvrez votre terminal et utilisez les mêmes commandes. Un client SSH est normalement installé par défaut.

Authentification SSH par clés

ssh sur Ubuntu

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 : http://www.it-connect.fr/les-cles-asymetriques/ 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.

Comment se connecter en SSH sans mot de passe?

https://tutox.fr/2017/04/08/se-connecter-ssh-de-passe/

Le grand classique du « comment faire communiquer 2 machines en SSH sans mot de passe »? Souvent on a besoin pour des scripts d’automatisation de s’affranchir de l’authentification par mot de passe entre 2 machines. Le SSH permet d’assurer la l‘authentification, la confidentialité des échanges entre 2 machines. C’est à dire que tout ce qui passe dans les tuyaux entre les 2 machines est chiffré.

Comment SSH fait il?

Il s’appuie sur la cryptographie asymétrique. Un nom bien pompeux derrière lequel se cache un mécanisme simple et rudement efficace.

Le principe:

On va reprendre le fameux exemple du « bob et alice ».

BOB souhaite parler à ALICE de manière confidentielle. Pour ce faire, ils disposent chacun de 2 clés:

- 1 clé privée et 1 clé publique.
- ALICE envoie en clair sa clé publique à BOB . Au passage, comme son nom l'indique , n'importe qui peut avoir accès à la clé publique d'ALICE. N'importe qui souhaitant communiquer avec elle utilise la clé publique d'ALICE.

La clé publique d'ALICE ,c'est en quelque sorte comme un cadenas que BOB va apposer sur son message.

chiffrement asymetrique

Ok , maintenant que Bob possède la clé publique d'ALICE, il va pouvoir lui écrire et chiffrer son message . (mettre son petit cadenas dessus : –)

Seule, ALICE avec sa clé privée pourra déchiffrer le contenu du message de BOB. La clé privée est en quelque sorte la clé qui ouvre le cadenas posé par BOB.

Scenario inverse:

- Alice veut répondre à BOB.
- Alice ⇒demande la clé publique de BOB
- BOB⇒envoie en clair sa clé publique à ALICE
- ALICE chiffre son message avec la clé publique de BOB
- BOB recoit le message et le déchiffre avec sa clé privée.

Concrètement, les commandes...

On veut que la machine A se connecte sur la machine B sans mot de passe.

1/ Générer les clés publiques (sur la machine A)

ssh-keygen -t rsa

Répondre « oui » et « rien » aux questions qui vous seront posées.

2 fichiers sont générés dans le répertoire « ~/.ssh »

- id_rsa ⇒ c'est la clé privée. A conserver précieusement à l'abri des indiscrets sur sa machine.
- id_rsa.pub ⇒ c'est la clé publique. Que l'on peut transmettre à tous ceux qui veulent communiquer avec moi.

2/ Copier la clé publique de la machine A

ssh-copy-id user@machineB

Vous devriez voir ces messages:

/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"

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

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

new keys

Rentrer le mot de passe habituel pour accéder à B.

Cette commande a pour effet de copier le contenu de la clé publique de A dans le fichier « ~/.ssh/authorized_keys » de B.

3/ Tester la connexion de A vers B

ssh user@machineB

et là magie magie, vous êtes directement connecté sur la machine B.

NOTA BENE:

Si c'est la toute première connexion de A vers B alors un message du type: The authenticity of host 'machineB (192.168.2.3)' can't be established. RSA key fingerprint is c6:ee:c6:e4:9a:b6:7e:46:4c:17:b4:d0:7b:80:af:2c. Are you sure you want to continue connecting (yes/no)? Répondre « yes » , l'empreinte la machine B sera copié dans le fichier « ~/.ssh/known_hosts » de la machine A. 4/ pour les curieux

Sur la machine B:

cat /home/user/.ssh/authorized_keys

on retrouve la clé publique de la machine A:

ssh-rsaAAAAB3NzaC1yc2EAAAADAQABAAABAQDIAePqXozbqzerGr3jCj7nF2Vd2Bqi7BbQJXw/ROGH+210Y41kEe9gzioV/Q86z31pMJdUhC+LU96wiBPbPMUlPqSNBckcbmQK0LoWZYaxwrlFdqcmbAhqx9lHywPCQdJCIGkkk9sFsuSmxli7mfuzNjKVdA6C8BuGvEgM1QBWQ9gGPIKCs+BXkCtfU1KyxZOdNnnP9apT/i2P+Pk3XocJJuXSR887Vu81a/hG2uy46/M8t/n7/Btrt44FZZ root@machineA

Si on supprimait cette clé du fichier authorized_keys de la machine B, à la prochaine tentative de connexion de A vers B, un mot de passe serait demandé.

Remarque:

```
  Pour être plus précis, le protocole SSH utilise le chiffrement asymétrique pour que client et serveur s'échange une clé de session.(je simplifie au max et ne parle pas de toutes les étapes pour ne pas embrouiller). Cette clé de session va permettre ensuite de chiffrer le reste des communications entre les 2 machines.
```

5/ sécurité?

- ne pas créer de connexions ssh automatique sans mot de passe pour Root.
- Conserver sa clé privée comme la prunelle de ses yeux . Au besoin la sauvegarder dans un lieu sûr.
- Du coup ,mettre des droits restrictifs sur la clé privée id_rsa. (par exemple : chmod 600 id_rsa)