administration sécurité serveur ssh

Ce document ne traite traite pas le fonctionnement du serveur SSH de windows

SSH

Cette page présente les usages les plus courants de SSH et sa configuration de base.
Voir sur SSH Avancé pour les autres usages.

SSH (https://fr.wikipedia.org/wiki/Secure_Shell) est un protocole permettant d'établir une communication chiffrée, donc sécurisée (on parle parfois de *tunnel*), sur un réseau informatique (intranet ou Internet) entre une machine locale (le *client*) et une machine distante (le *serveur*).

La sécurité du chiffrement peut être assurée par différentes méthodes, entre autre par mot de passe ou par un système de clés publique / privée (mieux sécurisé, on parle alors de cryptographie asymétrique (https://fr.wikipedia.org/wiki/cryptographie asymétrique)).

SSH remplace de manière sécurisée :

- Telnet : vous pouvez exécuter des commandes depuis un réseau local ou Internet via SSH
- FTP () : si vous ne souhaitez qu'ajouter ou modifier des fichiers sur un serveur, SFTP est bien plus adapté que FTP ()
- Et d'autres, via le « tunneling » : on peut accéder à un service réseau en le faisant circuler dans un tunnel SSH pour profiter de toutes les protections qu'il apporte. Vous pouvez donc sécuriser n'importe quel service grâce à SSH, comme VNC () par exemple.

Beaucoup d'utilisateurs de Telnet, Rlogin, FTP (), ou autres programmes de communication ne se rendent pas compte que leurs données et notamment les mots de passe sont transmis en clair sur le réseau, ce qui constitue une faille de sécurité évidente (voir *sniffing (https://fr.wikipedia.org/wiki/Analyseur_de_paquets)*).

Les usages de SSH sont entre autre :

- Accéder à distance à la console en ligne commande (shell), ce qui permet d'effectuer la totalité des opérations courantes et/ou d'administration sur la machine distante.
- Déporter l'affichage graphique de la machine distante.
- Transferts de fichiers en ligne de commande.
- Montage ponctuel de répertoires distants, soit en ligne de commande, soit via Nautilus sous GNOME par exemple.
- Montage automatique de répertoires distants.
- Déporter de nombreux autres services ou protocoles (voir SSH Avancé).

OpenSSH est la solution la plus utilisée pour mettre en place une communication SSH via un ensemble d'outils libres dont certains sont installés par défaut sur Ubuntu. Comme son nom l'indique, **OpenSSH** est développé dans le cadre du projet OpenBSD (http://www.openbsd.org). C'est cette solution que nous traiterons sur cette page.

1. Installation

Si vous voulez accéder à un ordinateur (votre ordinateur personnel, votre serveur local, un serveur distant dont vous effectuez l'administration, etc.). Vous devez installer **openssh-server (apt://openssh-server)** sur la machine à joindre en SSH, cette machine sera le "serveur" SSH.

La partie cliente est fournie par le paquet openssh-client (apt://openssh-client), qui est installé par défaut sous Ubuntu.

Vous pouvez vérifier ce qui est déjà installé en tapant ces commandes :

```
ssh -V
```

qui retourne une ligne du type :

```
OpenSSH_8.2p1 Ubuntu-4ubuntu0.5, OpenSSL 1.1.1f  31 Mar 2020
```

et aussi la commande suivante pour connaître la version de la bibliothèque ssl:

```
dpkg -l libssl*
```

1.1 Installation du serveur SSH

Installez le paquet **openssh-server (apt://openssh-server)** sur votre poste.

1.1.1 Utilisation du serveur SSH

Le serveur SSH fonctionne en tant que service lancé automatiquement au démarrage de la machine.
Il est possible notamment de :

- L'activer ou l'arrêter : par exemple si vous souhaitez désactiver momentanément le serveur SSH
- Le relancer : par exemple si vous faites une modification de configuration

Vous trouverez en fin de cette page plus d'information sur la Configuration du serveur SSH, peu sécurisée par défaut.

1.1.1.1 Vérifier le status

Saisissez dans un terminal la commande suivante :

```
sudo systemctl status ssh
```

1.1.1.2 Activer

Saisissez dans un terminal la commande suivante :

```
sudo systemctl start ssh
```

1.1.1.3 Arrêter

```
sudo systemctl stop ssh
```

1.1.1.4 Relancer

```
sudo systemctl restart ssh
```

1.2 Installation du client SSH

Sur le poste client **openssh-client** est déjà installé par défaut. Dans le cas contraire Installez le paquet **openssh-client (apt://openssh-client)**.

1.2.0.5 Clients pour machines qui ne sont pas sous Linux

Sous windows10 vous pouvez installer une Ubuntu via WSL (https://fr.wikipedia.org/wiki/Windows_Subsystem_for_Linux), vous aurez alors accès à un vrai client SSH :)

Si vous devez prendre le contrôle depuis un poste équipé de Windows vous pouvez installer PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/) qui est disponible sous licence MIT (une licence libre comparable à la licence BSD ()), openssh est installable dans le shell Windows depuis la version 1809 https://docs.microsoft.com/fr-fr/windows-server/administration/openssh/openssh_install_firstuse (https://docs.microsoft.com/fr-fr/windows-server/administration/openssh/openssh_install_firstuse).

Si vous voulez établir une connexion SSH depuis un smartphone Blackberry®, vous pouvez installer bbssh (http://bbssh.org/), qui est sous licence libre GPLv2, les sources sont fournies.

Il existe aussi des clients SSH pour Android (connectbot), J2ME (téléphones portables Java), iPhone, et quasiment tous les systèmes d'exploitation trouvables.

Vérifiez bien que UFW, le gestionnaire de firewall standard sous Ubuntu, n'est pas actif sur le serveur SSH **AVANT** l'installation de SSH. Il ne devrait pas fonctionner si vous ne l'avez pas activé.
Si UFW est actif, vous avez intérêt à vérifier s'il laisse passer le port standard du protocole SSH, le 22. Si ce n'est pas le cas, vous ne pourrez pas utiliser SSH sur cette machine.
Voyez la page UFW pour connaître le fonctionnement du pare-feu, ou, utilisez Gufw l'interface graphique du pare-feu UFW.

2. Utilisations de SSH

2.1 Accès à distance à la console en ligne de commande (shell ssh)

Pour ouvrir une session distant ayant un serveur SSH, vous devez écrire quelque chose comme ceci :

```
ssh <nom_utilisateur>@<ipaddress> -p <num_port>
```

Exemple :

```
ssh phyrex@192.168.23.42 -p 12345
```

L'option **-p <num_port>** qui précise le port utilisé par le serveur est facultative. Si rien n'est précisé, c'est le port 22 qui sera utilisé par défaut (protocole TCP ()).

Pour se connecter avec SSH en IPV6 () depuis un terminal, écrire sans crochet :

```
ssh -6 <nom_utilisateur>@<adresse ipv6>
```

Exemple : par exemple pour un lien Internet :

```
ssh -6 alfred@2a01:e35:2431::2e57
```

Vous pouvez aussi appeler un ordinateur par son nom :

```
ssh utilisateur@nom_machine
```

à partir du moment où celui-ci est résolu en adresse IP par votre machine.

Cela peut se faire sur le réseau local par le fichier **/etc/hosts** (ou bien, pour passer par une interface graphique, en tapant dans un terminal

```
network-admin
```

puis en allant dans l'onglet "Hôtes", continuer en déverrouillant les droits administration en cliquant sur le cadenas, et enfin, en cliquant sur le bouton "ajouter"), éventuellement distribué d'un serveur vers les clients locaux au travers de NIS (https://fr.wikipedia.org/wiki/Network_Information_Service), ou bien par un service de DNS (https://fr.wikipedia.org/wiki/Domain_Name_System) si vous accédez à une machine distante (serveur loué) pour lequel vous avez enregistré un nom de domaine.

Si vous voulez vous connecter à plusieurs machines situées derrière un routeur vous pouvez configurer celui-ci afin qu'il redirige chaque port TCP () entrant vers une machine donnée.
Exemple :
port externe 22001 redirigé vers 192.168.0.1:22
port externe 22002 redirigé vers 192.168.0.2:22
Ensuite utilisez l'option -p 22002 pour connecter par exemple la machine ayant pour adresse 192.168.0.2 sur le réseau local

2.2 Outil graphique pour les connexions SSH

Remmina est actuellement l'outil le plus complet pour la gestion des bureaux à distance.

2.3 Affichage graphique déporté (Tunneling serveurX par ssh) - Accéder aux applications graphiques

La commande ssh offre une fonctionnalité intéressante: la possibilité d'exécuter des applications X-Window à distance, ce qui est bien pratique pour travailler à distance, partager une machine ou simplement effectuer des tâches d'administration.

Il suffit de passer l'option -X à ssh :

```
ssh -X nomtilisateur@Ipserver
```

L'option **-X** permet le déport d'applications X-Window (affichage graphique à distance). Cette possibilité est offerte grâce aux fonctions de tunneling réseau présentes dans SSH.

N'oubliez pas qu'Ubuntu (Unix en général) est un système d'exploitation client/serveur, ce qui s'applique aussi à l'affichage géré par X-Window.

Et on peut lancer :

```
nomUtilisateur@Ipserver$ xeyes
```

là, l'œil de Moscou vous regarde depuis votre écran local !
N.B. : pour que cet exemple fonctionne il faut que le paquet *x11-apps* qui fournit *xeyes* soit installé sur le serveur.

Pour que vous puissiez afficher des applications X11 via SSH il faut que votre serveur ait la fonction "X11Forwarding" activée et que xauth y soit installé ! Allez À la fin de cette page pour savoir comment configurer votre serveur SSH.

2.4 Transfert - copie de fichiers

Pour copier un fichier à partir d'un ordinateur sur un autre avec SSH, vous devrez utiliser la commande **scp** ou **rcp**. Cela ressemblera à ceci :

```
scp <fichier> <username>@<ipaddressDistant>:<DestinationDirectory>
```

et en IPv6

```
scp -6 <élément> <nom>@[addresse ipv6]:<destination>
```

2.4.0.6 Exemples

Pour un fichier:

```
scp fichier.txt hornbeck@192.168.1.103:/home/hornbeck
```

et en IPv6

```
scp -6 fichier.txt albertine@[2a01:e35:2431::2a34]:/home/albertine
```

Pour un répertoire:

```
scp -r répertoire hornbeck@192.168.1.103:/home/hornbeck/
```

et en IPv6

```
scp -6r répertoire/ albertine@[2a01:e35:2431::2a34]:/home/albertine
```

Vous pouvez aussi bien copier des fichiers à partir des ordinateurs à distance sur votre disque local :

```
scp hornbeck@192.168.1.103:/home/hornbeck/urls.txt .
```

Ici, le point **.** à la fin de commande indique de copier le fichier dans le répertoire courant.

Pour les noms avec des espaces : de distant vers local

```
$ scp utilisateur@12.345.678.90:"le\ fichier" .
$ scp utilisateur@12.345.678.90:'"le fichier"' .
# Notez les simples ' ET doubles " guillemets ' "
```

```
$ fichier="le fichier" #ou
$ fichier=le\ fichier
$ scp utilisateur@12.345.678.90:"'$fichier'" .

$ fichier="le\ fichier"
$ scp utilisateur@12.345.678.90:"$fichier" .
```

Pour les noms avec des espaces :
de local vers distant c'est différent

```
$ scp  le\ fichier utilisateur@12.345.678.90:
le fichier                          100%    0     0.0KB/s   00:00

$ scp  "le fichier" utilisateur@12.345.678.90:
le fichier                          100%    0     0.0KB/s   00:00
```

```
$ fichier="le fichier"  # ou
$ fichier=le\ fichier
$ scp  "$fichier" utilisateur@12.345.678.90:
le fichier                          100%    0     0.0KB/s   00:00
```

cf : https://forum.ubuntu-fr.org/viewtopic.php?pid=21768296#p21768296 (https://forum.ubuntu-fr.org/viewtopic.php?pid=21768296#p21768296)

Vous pouvez aussi le renommer en le copiant (« mon.txt ») sur le disque local (toujours dans le répertoire courant):

```
scp hornbeck@192.168.1.103:/home/hornbeck/urls.txt ./mon.txt
```

Vous pouvez très bien copier un fichier d'un ordinateur vers un autre tout en étant sur un troisième ordinateur :

```
scp nom@ordi1:chemin/fichier nom@ordi2:chemin/fichier
```

Dans le cas où le port SSH du serveur ne serait pas le port par défaut (22), il faut indiquer le port distant à utiliser :

```
scp -P port fichier.txt hornbeck@192.168.1.103:/home/hornbeck
```

Lorsque l'on copie des fichiers ou des répertoires sur d'autres machines, ne pas oublier que les fichiers ou répertoires deviendront propriété du compte avec lequel on se connecte à distance. Pour préserver les propriétaire et groupe de chaque fichier ou répertoire, il sera donc utile de recourir à un logiciel tel que tar pour enregistrer l'intégralité des informations relatives à ce que l'on transfère.

2.5 Transfert - synchronisation de répertoire

Vous pouvez sécuriser un répertoire en le dupliquant à travers le réseau en utilisant la commande rsync.

Exemple1: Pour transférer le home d'un utilisateur **a** hébergé dans la machine **b** dont l'adresse IP est enregistrée dans le fichier /etc/hosts en excluant quelques fichiers avec un résultat à mettre dans le répertoire /MAISON qui sera automatiquement créé.

```
 sudo rsync -ahv --progress --stats --exclude={'.*','persistence.*'}  a@b:/home/a /
cat err.log
```

2.6 Monter un répertoire distant, navigation via SFTP (Secure File Transfer Protocol)

SFTP (https://fr.wikipedia.org/wiki/SSH_file_transfer_protocol) est une autre méthode pour accéder à ses fichiers via SSH. Au lieu de travailler fichier par fichier, il est possible grâce à cette méthode de naviguer dans ses fichiers depuis un client SFTP. Ce type d'accès est possible grâce à des outils comme Nautilus, Konqueror, Dolphin, WinSCP, Pcmanfm ou FileZilla, dont la mise en œuvre est décrite dans les sections suivantes.

2.6.1 Nautilus

En utilisant le gestionnaire de fichiers **Nautilus**, vous pouvez également accéder aux emplacements à distance par l'intermédiaire de SSH pour passer en revue, modifier et copier des fichiers.
Ouvrez Nautilus, puis dans la fenêtre emplacement (Ctrl + L), entrez l'URL () suivante en remplaçant `nom_utilisateur` , `hostname` et `port` en conséquence :

```
ssh://nom_utilisateur@hostname:port
```

La copie de fichier se fait avec le glisser-déposer dans la fenêtre de Nautilus comme sur votre système de fichiers local.

Pour accéder directement à un répertoire donné (pratique avec l'utilisation des signets), il suffit de rajouter le chemin en fin d'URL () :

```
ssh://username@hostname:port/le/chemin/voulu/
```

Il est également possible d'y avoir accès dans Nautilus par le menu *Fichiers → Autres emplacements → Se connecter à un serveur...* et choisir le `Type de service` « ssh ».

2.6.2 Dolphin ou Konqueror

Le principe est similaire à celui utilisé par Nautilus, à l'exception du nom de protocole : `sftp` .
Dans la barre d'adresse, tapez :

```
sftp://<nom_utilisateur>@<hostname>
```

Une boîte de dialogue apparaîtra et demandera le mot de passe.

Attention: Si vous ne mentionnez pas le nom d'utilisateur, c'est l'utilisateur courant sur la machine locale qui aura la main.

2.6.3 Dolphin

Le nouveau navigateur de KDE permet de faire ça très simplement.
Cliquez sur le raccourci `Réseau`, puis `Ajoutez un dossier réseau`. Remplissez ensuite les champs demandés. Pensez à mettre la racine (dossier **/**) comme dossier d'accès pour pouvoir rentrer sur l'intégralité de l'ordinateur distant.

Il est également possible de rentrer l'adresse et d'enregistrer le lien dans ses emplacements favoris :

```
sftp://nom_utilisateur@hostname:port
```

2.6.4 WinSCP

Parce qu'il est parfois nécessaire de faire un transfert de fichier à partir d'une machine sous MS Windows, il existe un logiciel libre nommé `WinSCP` qui permet de faire du SFTP avec une interface semblable à celle des clients FTP ().

Site officiel du logiciel WinSCP (https://winscp.net/eng/docs/lang:fr)

2.6.5 FileZilla

Sans avoir à chercher trop loin, FileZilla, le client FTP () compatible Linux, Windows et Mac OS () X, permet aussi la connexion à un serveur SFTP (SSH File Transfer Protocol) depuis la version 3.

2.6.6 PCManFM

Dans la barre d'adresse de PCManFM, rentrez ceci :

```
sftp://nom_utilisateur_distant@IPduSERVEUR/dossier/que/je/veux
```

2.7 Monter un répertoire distant de manière automatique (SFTP grâce à SSHFS)

SSHFS est un outil permettant d'utiliser le protocole **SSH** comme un système de fichier et ainsi monter un répertoire distant à travers le protocole SSH pour y accéder comme n'importe quel répertoire local à la manière d'un partage NFS; mais sécurisé !
Voir la page SSH Filesystem.

3. Authentification

Si vous ouvrez votre serveur SSH sur Internet, par exemple pour y accéder depuis l'ordinateur d'un ami(e) ou lui permettre d'accéder à certains de vos fichiers, n'oubliez JAMAIS qu'Internet est parcouru par des robots qui scannent et testent en permanence tous les serveurs disponibles (SSH et autres) et qu'ils vont faire des tentatives pour trouver vos mots de passe de compte (on parle d'attaque par force brute (https://fr.wikipedia.org/wiki/attaque par force brute)). L'usage des clés est donc très fortement recommandé.
Si vous ne pouvez vraiment pas faire autrement, utilisez des mots de passe longs et complexes ainsi qu'un système de protection tel que fail2ban qui permet de bannir des adresses IP au bout d'un certain nombre de tentatives erronées.
Voir aussi denyhosts notamment en cas de:

```
ssh_exchange_identification: read: Connection reset by peer
```

3.1 Authentification par mot de passe

L'authentification par mot de passe (transmis chiffré) est le mode d'identification par défaut.

3.2 Authentification par un système de clés publique/privée

3.2.1 Description

Autrefois tout le monde employait l'authentification typique par le principe *identifiant - mot de passe*. Cependant si quelqu'un connaît votre mot de passe ou le découvre au moyen d'une attaque la sécurité est compromise. De plus, utiliser un mot de passe différent pour chaque serveur et le saisir à chaque connexion peut s'avérer contraignant.

Pour se débarrasser des ces problèmes, SSH propose un système d'authentification par clé publique/privée au lieu des mots de passe « simples ».

Ceci peut permettre par exemple :

- à un administrateur de se connecter à des centaines de machines sans devoir connaître des centaines de mots de passe différents ;
- de ne pas avoir un mot de passe à saisir toutes les 2 minutes (en utilisant *ssh-agent*).

Ces clés restent des chaînes de caractères (en français courant : du texte), mais sont beaucoup plus longues et aléatoires que de simples mots de passe.

La clé privée est en principe unique : chaque utilisateur possède une clé privée qu'il peut copier sur les terminaux auxquels il accède physiquement et depuis lesquels il a besoin d'un accès SSH (via le client SSH). Cette clé se trouve généralement dans un fichier `~/.ssh/id_xxxx` . C'est un document sensible très personnel, il faut y appliquer des droits très restrictifs : `r-x — —` (500) pour le répertoire .ssh et `r– — —` (400) pour le fichier id_xxxx.
Pour un maximum de sécurité il est possible de protéger cette clé privée au moyen d'un mot de passe.

De son côté la clé publique est envoyée à tous les serveurs auxquels on veut accéder à distance afin qu'ils nous identifient avec certitude en vérifiant que le client possède bien la clé privée associée. Côté serveur cette clé publique sera stockée dans le fichier `~/.ssh/authorized_keys` , avec éventuellement des options et les clés publiques d'autres utilisateurs à raison d'une par ligne.

Localement on peut stocker une clé publique par ex. dans un fichier `id_xxxx.pub` qui peut aussi se trouver dans le répertoire `~/.ssh` , ou ailleurs (ce n'est pas un document sensible).

On peut générer une nouvelle clé publique depuis une clé privée mais pas l'inverse.

3.2.2 Mise en place des clés

À moins que vous n'ayez déjà un couple de clés, vous devez d'abord en créer.
Exemple pour une clé utilisant l'algorithme de chiffrement ED25519, vous saisirez dans le terminal du client :

```
ssh-keygen -t ed25519 -C "email@example.com"
```

On peut également utiliser l'algorithme plus ancien, moins sûr et générant des clés plus grosses RSA :

```
ssh-keygen -t rsa  -b 4096 -C "email@example.com"
```

Il vous sera alors demandé où enregistrer la clé privée (acceptez juste l'endroit par défaut : **~/.ssh**, et ne changez pas le nom du fichier généré) puis de choisir une *passphrase* (phrase de reconnaissance).

Bien que non obligatoire, l'utilisation d'une *passphrase* est recommandée pour protéger votre clé privée. En effet toute personne qui obtiendrait l'accès à votre clé privée (non protégée) aurait alors vos permissions sur d'autres ordinateurs. Veuillez prendre un instant et choisissez une très bonne *passphrase* c'est à dire longue et complexe.

Votre clef publique a été créée avec la nouvelle clef privée. Elles sont habituellement localisées dans le dossier caché **~/.ssh**:

~/.ssh/id_ed25519.pub pour la clé publique et **~/.ssh/id_ed25519** pour la clé privée ou **~/.ssh/id_ras.pub** et **~/.ssh/id_rsa** si vous avez utilisé l'algorithme RSA.

Pour que votre ssh-agent reconnaisse cette paire de clefs, il faut utiliser la commande ssh-add :

```
ssh-add
#ou plus directement
ssh-add /chemin-complet/vers-la-cle/nom-cle
```

Vous pouvez également vérifier la liste des paires de clefs existantes avec l'option -l (-list) :

```
ssh-add -l
```

Si cette commande ressort :

```
The agent has no identities.
```

alors aucune clef n'est actuellement prise en compte. Il faut recommencer les étapes ci-dessus.

Si vous avez :

```
Could not open a connection to your authentication agent.
```

alors faire

```
eval "$(ssh-agent)"
```

puis de nouveau

```
ssh-add
```

pistes pour débugger :

- forcer l'utilisation uniquement de clefs dans la commande ssh :

```
ssh -o PubkeyAuthentication=yes -o PreferredAuthentications=public
```

- utiliser les options **-v** ou **-vv** ou **-vvv** dans le commande ssh

Il faut maintenant envoyer au serveur votre clé publique pour qu'il puisse vous chiffrer des messages.

En résumé (car les paragraphes ci-dessous utilisant des scripts peuvent sembler confus à certains)

- Coté client : Il faut que le client ait mis sa clé privée en $HOME/.ssh/ (côté client).
- Coté client : la clé doit être connue de l'agent ssh (ssh-add)
- Coté client : Le répertoire $HOME/.ssh doit appartenir au propriétaire de $HOME et les clés privées ne doivent être accessibles que par leur propriétaire (mode rw------ ou 600 au maximum)
- Coté serveur : La clé publique du client doit se trouver dans le fichier $HOME/.ssh/authorized_keys du serveur.
- Coté serveur : il vaut mieux refuser l'accès par mot de passe ("PasswordAuthentication no" dans /etc/ssh/sshd_config du serveur)

L'utilisateur distant doit avoir cette clé (c'est une ligne de caractères en code ASCII ()) dans son fichier de clés d'autorisation situé à **~/.ssh/authorized_keys** sur le système distant. Employez la commande *ssh-copy-id*.

ssh-copy-id est un script qui utilise ssh pour se connecter à une machine à distance en utilisant le mot de passe de l'utilisateur. L'authentification par mot de passe doit donc être autorisée dans le fichier de configuration du serveur ssh (par défaut sur Ubuntu). Il change également les permissions des répertoires **~/.ssh** et **~/.ssh/authorized_keys** de l'hôte distant pour enlever l'accès en écriture du groupe (qui vous empêcherait de vous connecter si le serveur distant ssh a "StrictModes yes" dans son fichier de configuration, ce qui est le cas par défaut sur Ubuntu).

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub <username>@<ipaddress>
```

ou si le port est différent du port standard 22 (notez les guillemets (https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=99785)):

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub -p <num_port> "<username>@<ipaddress>"
```

Vous devrez alors donner le mot de passe *utilisateur* de cet ordinateur. Après l'ajout votre clé publique, vous devenez un hôte de confiance.

Si l'authentification par mot de passe est désactivée[1)], alors vous aurez besoin de copier-coller votre clé suivant un autre moyen.
Voici une ligne à copier pour ajouter sa clé publique sur le serveur distant :

```
ssh login@serveur "echo $(cat ~/.ssh/id_id_ed25519.pub) >> .ssh/authorized_keys"
```

Lancez :

```
ssh <username>@<ipaddress> -p <num_port>
```

Dorénavant n'utilisez plus votre mot de passe mais votre **passphrase** pour vous connecter. Celle-ci sert à déchiffrer votre *clé privée* de votre système local.

Si ça ne marche pas, c'est-à-dire que le mot de passe vous est quand même demandé, essayez sur votre serveur la commande :

```
tail -f /var/log/auth.log
```

tandis que vous essayez de vous connecter. Si on vous parle de "*vulnkey*", c'est que par malchance `ssh-keygen` a généré une clé vulnérable. Recommencez alors la procédure depuis le début[2)]

Pour résumer, deux choses sont nécessaires pour obtenir un accès réellement sécurisant (et sécurisé 😉) par authentification à clé publique par rapport à l'authentification par mot de passe classique :

1. **Votre clé privée**, chiffrée ;
2. **Votre passphrase**, utilisée pour déchiffrer votre clé privée.

Si vous choisissez de ne pas avoir de mot de passe (ce qui est possible, voyez la prochaine section), vous aurez une sécurité moindre, ainsi que si vous utilisez une authentification uniquement par mot de passe, comparé à celle que vous pouvez avoir en combinant les deux.

3.2.3 Éléments importants en lien avec l'usage des clés

3.2.3.7 Authentification par mot de passe et / ou par clé

Vous pouvez avoir avec SSH les deux modes d'authentifications actifs en même temps, par mot de passe et par clés.

Vous pouvez vouloir neutraliser l'authentification par mot de passe pour des raisons de sécurité, pour cela il faut modifier le fichier de configuration **/etc/ssh/sshd_config** de la manière suivante :

- A la ligne `PasswordAuthentication` mettre `no`

N'oubliez pas de relancer le service ssh sur votre serveur après avoir changé la configuration.

3.2.3.8 Vulnérabilité des anciennes clés

Au mois de mai 2008 a été découvert une faiblesse dans la génération des clés par OpenSSL des packages Debian et dérivés tels qu'Ubuntu. Pour résumer, si vous avez généré vos clés entre 2006 et mai 2008, il faut en créer de nouvelles après avoir mis à jour le système. Pensez alors à bien rediffuser vos clés.

3.2.3.9 Mot de passe toujours demandés avec authentification par clés

Si, après avoir suivi ce tutoriel, un mot de passe est toujours demandé, il se peut que ce soit dû à un problème de droits sur votre *Dossier Personnel*.
Sur la machine distante regardez le fichier **/var/log/auth.log** pour y trouver des indications et notamment si la ligne suivante apparaît :

```
Authentication refused: bad ownership or modes for directory /home/votre_login
```

Alors faites :

```
chmod 755 $HOME
```

Et tout devrait rentrer dans l'ordre.

Si ce n'est toujours pas le cas, c'est que le serveur doit être configuré en mode de sécurité strict (c'est le cas par défaut sur Ubuntu).
Effectuez les opérations suivantes :

Sur le serveur :

- dans le fichier **/etc/ssh/sshd_config**, la ligne `StrictModes yes` indique que le serveur va être très pointilleux sur les droits du compte sur lequel on se connecte en ssh.
- saisissez ensuite les commandes suivantes

```
chmod go-w ~/
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

- Sur le client, dans **/etc/ssh/ssh_config**, rajoutez la ligne `PreferredAuthentications publickey`.

3.2.3.10 Gestion des clés

Parfois les clés de vos correspondants peuvent changer (réinstallation de machine par exemple), vous aurez alors droit à ce charmant message :

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx.
Please contact your system administrator.
Add correct host key in /home/<vous>/.ssh/known_hosts to get rid of this message.
Offending key in /home/<vous>/.ssh/known_hosts:4
RSA host key for <ip> has changed and you have requested strict checking.
Host key verification failed.
```

Soit l'information est exacte et une machine a été corrompue, ou bien il s'agit juste d'un changement de clé (réinstallation par exemple) et dans ce cas il faut effacer les entrées dans le fichier **.ssh/known_hosts** de votre compte.
Avant la chose était relativement simple, la clé était directement associée au nom ou à l'IP de la machine cible. Ce n'est plus le cas à présent où elle est associée par UUID rendant quasiment impossible l'identification visuelle de la ligne concernée. Mais ssh étant sympathique, il vous indique quelle est la ligne du fichier concernée.
Pour reprendre l'exemple précédent on peut lire la ligne Offending key in /home/<vous>/.ssh/known_hosts:4 → la clé en erreur est donc située ligne 4 du fichier **.ssh/known_hosts**

Il existe cependant une méthode plus subtile en employant la commande suivante :

```
ssh-keygen -R <ip>
```

Vous pourrez ainsi effacer seulement l'adresse IP concernée et relancer un ssh.

3.2.3.11 Connexion à un répertoire /home chiffré

Si vous souhaitez vous connecter par SSH avec une clef publique sur un compte dont le home est chiffré, il est important de faire attention à ce que sur le serveur le fichier/dossier **.ssh/authorized_keys** soit à la fois dans le home chiffré et déchiffré. En effet si le fichier authorized_keys est dans le home sous forme chiffré (.private), open_ssh ne pourra pas lire la clef publique attendue. Il faut donc créer un dossier .ssh et y mettre le fichier authorized_keys quand le

home est démonté donc chiffré. Cependant, si vous ne le laissez pas aussi dans le home déchiffré et donc monté, la connexion SSH se fera avec la clef publique mais le home ne sera pas déchiffré automatiquement.

La meilleure solution est de créer des liens virtuels vers un dossier qui n'est pas soumis au chiffrement/déchiffrement comme expliqué ici (https://rohieb.wordpress.com/2010/10/09/84/)

3.2.3.12 Authentification SSH avec plusieurs clés privées

En principe chaque utilisateur possède une clé privée unique qui lui est propre, et envoie sa clé publique à autant de serveurs qu'il le souhaite. Cependant il est techniquement faisable de posséder plusieurs clés privées (mais c'est moins pratique et ça n'est pas plus sécurisé).

Lorsqu'on se connecte à plusieurs serveurs avec des clés privées différentes, il faut pouvoir indiquer à SSH quelle clé on veut utiliser pour la connexion, sinon, on rencontre par exemple l'erreur:

```
git@gitlab.com: Permission denied (publickey).
```

Pour indiquer au client SSH la clé qu'il doit utiliser pour chacun des serveurs on peut créer un fichier **~/.ssh/config** (ou **/etc/ssh/ssh_config** pour tous les utilisateurs de la machine) dans lequel il faut spécifier pour chacun des serveurs la clé qui doit être utilisée :

```
Host adresse-serveur-sans-passphrase.com
User votreutilisateur
IdentityFile ~/.ssh/key-sans-passphrase

Host adresse-serveur-avec-passphrase.com
User votreutilisateur
IdentityFile ~/.ssh/key-avec-passphrase
```

Pour plus d'options, comme l'utilisateur ou le port à utiliser par défaut, voir le manuel de **ssh_config**, cf. aussi l'aide de gitlab (en) (https://gitlab.com/help/ssh/README#working-with-non-default-ssh-key-pair-paths)

3.2.3.13 Les empreintes (fingerprint)

Retrouver l'empreinte de notre clef SSH, pour la communiquer à une personne qui veut se connecter et va utiliser notre clef publique :

```
ssh-keygen -l
```

Ensuite la commande demande le fichier de la clef publique. Sur un serveur on spécifiera **/etc/ssh/ssh_host_rsa_key.pub**.

4. Configuration du serveur SSH

La configuration par défaut du serveur SSH sous Ubuntu est suffisante pour fonctionner correctement. Le fichier de configuration à éditer avec les droits d'administration est **/etc/ssh/sshd_config**.

Tableau des principales directives à modifier le cas échéant :

Directive du fichier	Valeur par défaut sous Ubuntu	Valeur possible	Effet de la valeur choisie
Port	22	Tous les ports qui ne sont pas utilisés par un autre service	Le changement du port par défaut , souven la sécurité. C'est même l'inverse si vous cho c'est à dire supérieur à 1024
PermitRootLogin	without-password / prohibit-password	yes no without-password forced-commands-only prohibit-password	cf le man (http://manpages.ubuntu.com/manpages/l
PubkeyAuthentication	yes	no	Laisser yes si vous voulez établir l'authentif expliqué plus haut
PasswordAuthentication	yes	no	On peut parfaitement conserver l'authentif utilisateurs avec celle par mot de passe pou Conserver cette valeur à yes tant que l'auth pleinement fonctionnelle, sinon vous perdr
X11Forwarding	yes	no	Laisser yes pour faire de l'affichage graphic
#MaxStartups 10:30:60	ligne commentée donc inactive	décommenter (enlever symbole #)	Le 10 représente le nombre de connexions utilisateur ait réussi à s'identifier, si cela pa de probalités que les suivantes soient bloq augmente linéairement jusqu'à 100 % lorsc connexions. Très utile pour éviter ce genre (http://linuxfr.org/~dark_star/18379.html)
#Banner /etc/issue.net	Ligne commentée donc inactive	Décommenter	Lorsque vous essayez de vous connecter à v fichier `/etc/issue.net` est affiché (à vous bonjour ou mettre un avertissement, un gu
UsePAM	yes	no	Attention, bien lire la page de man sshd_co avec *ChallengeResponseAuthentication* et
AllowUsers	Ligne absente (autorisé à tous)	ajouter la ligne avec valeur(s) : AllowUsers Alice Bob	Spécifie les *logins* des seuls utilisateurs aut *pour ouvrir un compte FTP () à un ami tout via SSH*.
DenyUsers	Ligne absente (interdit à personne)	Ajouter la ligne avec valeur(s)	Interdit l'accès à SSH aux utilisateurs listés
AllowGroups	Ligne absente (autorisé à tous les groupes)	ajouter la ligne avec valeur(s) : AllowGroups groupname1 groupname2	L'authentification via SSH ne sera possible groupes désigné par leur nom (pas par GID)

Directive du fichier	Valeur par défaut sous Ubuntu	Valeur possible	Effet de la valeur choisie
DenyGroups	Ligne absente (interdit à aucun groupe)	Ajouter la ligne avec valeur(s)	Interdit l'accès à SSH aux utilisateurs des gr
ClientAliveInterval	Ligne absente	Ajouter la ligne avec valeur en secondes : ClientAliveInterval 300	Permet dans certains cas de maintenir une

Pour plus d'informations consultez `man sshd_config` .

5. Configuration du client SSH

Le client peut aussi être configuré… via le fichier **/etc/ssh/ssh_config** Il est notamment intéressant d'ajouter

```
        ServerAliveInterval 240
```

Pour que le client envoie des infos au server (ici toute les 4 minutes) et que ce dernier ne coupe pas la liaison, ce qui bloque (fige) le terminal. Pour voir les autres options: https://man.openbsd.org/ssh_config (https://man.openbsd.org/ssh_config) (en)

6. Accéder à un serveur SSH dont les ports entrants sont bloqués

Il peut arriver que les ports des connexions entrantes sur un serveur SSH soient bloqués [3)]. Cependant, il est rare que les ports sortants soient fermés. Dans ce cas, il est possible de faire appel à du « *Reverse-SSH* » tel qu'expliqué dans **cette page**

7. Voir aussi

- site officiel (https://www.openssh.com/)
- cssh : Cluster SSH
- note ministérielle du 17 août 2015 (https://www.ssi.gouv.fr/uploads/2014/01/NT_OpenSSH.pdf) : Recommandations pour un usage sécurisé d'(Open)SSH

- Guide de Mozilla pour la configuration d'OpenSSH (https://infosec.mozilla.org/guidelines/openssh)
- Tutoriels sur l'utilisation et la configuration avancée de SSH (https://www.it-connect.fr/cours-tutoriels/administration-systemes/linux/ssh/) sur IT-Connect
- Guide de ssh-audit pour blinder la configuration (https://www.ssh-audit.com/hardening_guides.html)
- ssh_avance Fixme !

Contributeurs: sx1, krodelabestiole, Zer00CooL, bruno

---

[1] donc PasswordAuthentication **no** dans **/etc/ssh/sshd_config** sur le serveur
[2] donc à partir de `ssh-keygen`
[3] le cas peut se présenter notamment en entreprise ou derrière une box

ssh.txt Dernière modification: Le 25/11/2022, 17:45 par PhiX