Linuxtricks.fr Site mémo et blog Table des matières

Linuxtricks

Recherche...

- Loi n° 78-17 d Table des matières
- Sondages
- Faire un don

Menu d'actions du module Wiki
- Accueil
- Explorateur

1. Accueil
2. Wiki
3. Services et serveurs
4. SSH : Installer et configurer un serveur SSH

Services et serveurs

SSH : Installer et configurer un serveur SSH

Outils
- Historique
- Version imprimable

Qu'est ce que SSH et OpenSSH

Secure Shell (SSH) est un programme mais aussi un protocole de communication sécurisé. Grâce à SSH, on peut se connecter à distance sur une machine, transférer des fichiers, si celle-ci dispose d'un serveur SSH,

OpenSSH (OpenBSD Secure Shell) est un ensemble d'outils informatiques libres permettant des communications sécurisées sur un réseau informatique en utilisant le protocole SSH.

Installation du serveur OpenSSH

Pour installer le serveur ssh sur notre machine, installer le paquet openssh-server

Table des matières

Gentoo et Calculate Linux :

Code BASH : **Copier vers le presse-papier**

```
emerge net-misc/openssh
```

CentOS et Fedora :

Code BASH : **Copier vers le presse-papier**

```
dnf install openssh-server
```

CentOS 7 et 6 :

Code BASH : **Copier vers le presse-papier**

```
yum install openssh-server
```

Debian :

Code BASH : **Copier vers le presse-papier**

```
apt install openssh-server
```

Configurer le service SSH

Lancer le service au démarrage

Gentoo (OpenRC) / Calculate Linux :

Code BASH : **Copier vers le presse-papier**

```
rc-update add sshd default
/etc/init.d/sshd start
```

Fedora et CentOS :

Table des matières

Code BASH : **Copier vers le presse-papier**

```
systemctl enable --now sshd.service
```

Debian :

Code BASH : **Copier vers le presse-papier**

```
systemctl enable --now ssh.service
```

Affiner la configuration de SSH

Désactiver les connexions SSH en root

Il est **recommandé** de ne pas permettre la connexion en root.

Pour cela, éditer le fichier **/etc/ssh/sshd_config** et modifier cette ligne:

Code : **Copier vers le presse-papier**

```
PermitRootLogin no
```

Si on souhaite se connecter en root directement, on peut n'autoriser que la connexion avec clé :

Code : **Copier vers le presse-papier**

```
PermitRootLogin prohibit-password
```

Ne pas oublier de redémarrer SSH.

Changer le port de SSH

Je vous recommande de changer le port.

Par exemple, utiliser le port 2222, éditer /etc/ssh/sshd_config :

Table des matières

Code BASH : **Copier vers le presse-papier**

```
Port 2222
```

Ne pas oublier d'ajouter une exception pour le parefeu Exemple avec firewalld :

Code BASH : **Copier vers le presse-papier**

```
firewall-cmd --add-port=2222/tcp --permanent
firewall-cmd --reload
```

Si on utilise SELinux, on doit changer le booléen concerné :

Code BASH : **Copier vers le presse-papier**

```
semanage port -a -t ssh_port_t -p tcp 2222
```

Redémarrer SSH pour prendre effet.

Autoriser le SSHFS

Il se peut que lorsque vous tentez de vous connecter en SSHFS sur votre seveur, celui-ci indique **Connexion reset by peer**.

Pour cela, éditer le fichier **/etc/ssh/sshd_config** et modifier/ajouter cette ligne:

Sous Gentoo :

Code : **Copier vers le presse-papier**

```
Subsystem       sftp    /usr/lib64/misc/sftp-server
```

Chez CentOS, cela ressemble à

Code : **Copier vers le presse-papier**

```
Subsystem    sftp    /usr/libexec/openssh/sftp-server
```

Ne pas oublier de relancer SSH

Table des matières

Autoriser/Interdire des utilisateurs

Il est possible d'autoriser une liste de certains utilisateurs à se connecter. Modifier ou ajouter cette ligne dans le sshd_config :

Code BASH : **Copier vers le presse-papier**

```
AllowUsers user1 user2 user3
```

Il est possible de refuser la connexion que de certains utilisateurs. Modifier ou ajouter cette ligne dans le sshd_config :

Code BASH : **Copier vers le presse-papier**

```
DenyUsers user1 user2 user3
```

Autoriser/Interdire des groupes

Il est possible d'autoriser une liste de certains groupes à se connecter. Modifier ou ajouter cette ligne dans le sshd_config :

Code BASH : **Copier vers le presse-papier**

```
AllowGroups groupe1 groupe2
```

Il est possible de refuser la connexion que de certains groupes. Modifier ou ajouter cette ligne dans le sshd_config :

Code BASH : **Copier vers le presse-papier**

```
DenyUsers groupe1 groupe2
```

Afficher une bannière

Si le serveur est public, on peut ajouter une bannière, une sorte de message affiché à la connexion.

Exemple : Table des matières

Code BASH : **Copier vers le presse-papier**

```
################################################################################
#                                                                              #
#  WARNING : Unauthorized access to this system is forbidden and will be      #
#  prosecuted by law. By accessing this system, you agree that your actions   #
#  may be monitored if unauthorized usage is suspected.                        #
#                                                                              #
################################################################################
```

On décommente la ligne "Banner" dans le fichier **/etc/ssh/sshd_config** et on y place le nom du fichier de la bannière :

Code BASH : **Copier vers le presse-papier**

```
Banner /etc/banner
```

On créé le fichier **/etc/banner** avec notre bannière dedans

Directive Match

Si on souhaite encore personnaliser certaines options dans SSh pour certains groupes, utilisateurs ou adresses, on peut utiliser la directive Match :

Voici deux exemples.

Autoriser l'accès root en fonction de certaines adresses :

Code BASH : **Copier vers le presse-papier**

```
Match Address 192.168.21.0/24
    PermitRootLogin yes

Match Address *
    PermitRootLogin no
```

Ou bien n'autoriser que quelques options pour un groupe particulier (CF Tuto SFTP) :

Code BASH : **Copier vers le presse-papier**

```
Match Group sftpusers
    ChrootDirectory /sftp/%u
    AllowTcpForwarding no
    ForceCommand internal-sftp
```

Table des matières

Ne pas oublier le parefeu

Bien qu'automatiquement, lors de l'installation d'openssh, celui-ci ouvre le parefeu, vérifier, selon la distribution utilisée que le parefeu est bien ouvert sur le port **22** en **TCP**.

Avec firewalld :

Code BASH : **Copier vers le presse-papier**

```
firewall-cmd --add-service=ssh --permanent
firewall-cmd --reload
```

Tester sa connexion ssh :

En local

Ouvrir une console, et tester de se connecter (dans mon exemple, avec l'utilisateur adrien) :

Code BASH : **Copier vers le presse-papier**

```
adrien@masupermachine ~ $ ssh adrien@127.0.0.1
Password:
Last login: Thu Jan 24 20:50:28 CET 2013 from 192.168.1.200 on pts/3
adrien@masupermachine ~ $
```

A distance

Ouvrir une console sur un autre PC (on considère que l'IP de ma machine où est installée SSH est 192.168.1.101 :

Code BASH : **Copier vers le presse-papier**

```
adrien@masupermachine ~ $ ssh adrien@192.168.1.101
Password:
Last login: Thu Jan 24 20:50:28 CET 2013 from 192.168.1.200 on pts/3
adrien@masupermachine ~ $
```

Table des matières

Si on a changé le port :

Code BASH : **Copier vers le presse-papier**

```
ssh adrien@192.168.1.101 -p2222
```

Depuis Windows, on pourra utiliser PuTTY ou Bash dans Windows 10.

Et si on parlait de clés ?

Le mot de passe pour se connecter en SSH, c'est bien mais ... Si on se connectait avec une clé que seul notre ordinateur et le serveur en ont connaissance ?

C'est une bonne alternative au mot de passe. Si le serveur est directement connecté à l'Internet, il est préférable, d'utiliser une paire de clés privées/publiques et d'interdire le mot de passe.

Mais cela peut être pratique, pour ne pas saisir de mot de passe et se connecter cash au serveur (automatismes rsync par exemple).

Générer la paire de clés

Dans l'exemple qui suit, je vais générer la clé au format RSA avec une taille de 4096 bits :

Code BASH : **Copier vers le presse-papier**

```
ssh-keygen -t rsa -b 4096
```

Il faut renseigner par la suite le nom du fichier, valider pour laisser par défaut dans **/home/$USER/.ssh/id_rsa**.

Ensuite, on nous demande de saisir une passphrase. C'est une suite de caractères (une phrase mnémotechnique, qu'il faudra saisir à chaque connexion). Je ne vais pas en mettre.

Ce n'est pas ANTI-SÉCURITÉ puisque sur la machine client ET le serveur devront s'authentifier mutuellement.

Pour plus de sécurité, on peut utiliser ECDSA au lieu de RSA avec une taille de 384 bits :

Table des matières

Code BASH : **Copier vers le presse-papier**

```
ssh-keygen -t ecdsa -b 384
```

Voire même on peut utiliser ed25519 :

Code BASH : **Copier vers le presse-papier**

```
ssh-keygen -t ed25519
```

ecdsa et **ed25519** : c'est de la cryptographie sur des courbes elliptiques. Retenez rapidement que c'est plus efficace et moins consommateur de ressources. Il y a plein d'infos sur le net à ce sujet.

Voici ce que contient le fichier de clé publique sur notre client:

Code TEXT : **Copier vers le presse-papier**

```
adrien@client ~ $ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3Nz[...]dR/7L adrien@client
```

Configurer le serveur

Et maintenant ? Comment je configure mon serveur ?

Et bien, la clé publique, précédemment générée est à copier coller sur le serveur, dans le fichier **~/.ssh/authorized_keys**.

Attention ! Le contenu du fichier ~/.ssh/id_rsa.pub doit tenir sur une seule et même ligne !

Pour éviter les erreurs de frappe, si le client est un Linux, on peut transférer le fichier via cette commande:

Code BASH : **Copier vers le presse-papier**

```
ssh-copy-id adrien@serveur
```

Sur notre serveur, nous obtenons ensuite ceci :

Table des matières

Code TEXT : **Copier vers le presse-papier**

```
adrien@serveur ~ $ cat .ssh/authorized_keys
ssh-rsa AAAB3N[...]oe1 adrien@client
```

Si plusieurs clients ont des accès au serveur, il y aura plusieurs lignes dans le fichier **~/.ssh/authorized_keys** du serveur.

Maintenant, on teste la connexion au serveur.

> Si c'est la première fois qu'on se connecte au serveur par ssh, avec notre client, il peut y avoir un avertissement. Valider par "Yes"
>
> Cf. exemple en dessous

Nous sommes connectés.

On ferme la connexion avec la commande **exit** et on tente de se connecter : RAS, rien à saisir !

Code TEXT : **Copier vers le presse-papier**

```
adrien@client ~ $ ssh adrien@192.168.1.101
The authenticity of host '192.168.1.101 (192.168.1.101)' can't be established.
ECDSA key fingerprint is d3:01:93:21:a4:37:24:d0:49:58:f2:57:67:7b:be:02.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.101' (ECDSA) to the list of known hosts.
Last login: Thu Jan 24 19:05:59 CET 2013 on :0
adrien@serveur ~ $ exit
déconnexion
Connection to 192.168.1.101 closed.
adrien@client ~ $ ssh adrien@192.168.1.101
Last login: Thu Jan 24 20:38:37 CET 2013 from 192.168.1.200 on pts/3
adrien@serveur ~ $
```

Et voila, notre client et notre serveur sont configurés.

Les déclinaisons des commandes ssh (telles rsync, scp et sftp) bénéficient du système de clés privées/clés publiques.

Si on est sur de notre coup, on peut désactiver les connexions par mot de passe à notre serveur pour n'autoriser que des connexions par clés.

Ne faites cela que si vous êtes sûr de votre coup, sinon, il vous faudra accéder à la machine physique si vous perdez la clé (vol d'ordinateur client, destruction du disque dur ...) et que vous n'avez pas d'autres clients configurés.

Pour cela, éditer le fichier **/etc/ssh/sshd_config** et passer la valeur

PasswordAuthentication à **no** Table des matières

Recharger la configuration de ssh.

Et si je ne veux plus qu'un client accède à mon serveur

Et bien la réponse à cette question est simple, il suffit de supprimer la ligne correspondante dans le fichier **~/.ssh/authorized_keys** du serveur.

Infos pratiques côté Client

~/.ssh/config : Configuration simple

Je vais vous présenter quelque chose qui m'a simplifié la vie : les alias SSH.

Si on a l'habitude de se connecter à plusieurs serveurs SSH, (noms d'hôtes différents, ports différents) et que vous avez parfois la *flemme* de taper toute la ligne :

Code BASH : **Copier vers le presse-papier**

```
ssh -p 5222 adrien@192.168.10.100
```

Il est possible de créer des alias.

Il suffit de créer le fichier **~/.ssh/config**.

Voici un exemple :

Code BASH : **Copier vers le presse-papier**

```
Host srv1
  HostName 192.168.10.100
  Port 5222
  User adrien
```

Ainsi, j'ai juste à taper en console **ssh srv1** et cela est équivalent à **ssh -p 5222 adrien@192.168.10.100**.

Table des matières

~/.ssh/config : Configuration avancée

Alias de tunnel

Il est même possible de créer un alias tunnel.

Je vous montre un exemple, puis je vous explique :

Code BASH : **Copier vers le presse-papier**

```
Host webmin
  HostName 192.168.1.11
  Port 22
  User adrien
  LocalForward 8080 192.168.1.11:10000
```

Quand je fais faire **ssh webmin**, ça va m'ouvrir un tunnel SSH entre mon PC et le serveur ayant l'IP 192.168.1.11.

On créé donc un tunnel SSH, qui va "relier" le port 8080 de ma machine sur le port 10000 de la machine distante.

Si le serveur n'a que le pot 22 d'ouvert, c'est intéressant. Car je peux accéder à webmin (qui écoute sur le port 10000) maintenant, avec le navigateur sur mon PC à l'adresse https://127.0.0.1:8080 .

Tunnels et redirection de ports

Redirection locale de port

L'option **-L** : *locale* pour rediriger un port distant vers une destination (machine) locale

La syntaxe de la commande :

Code BASH : **Copier vers le presse-papier**

```
ssh machine-distante -L port-local:HOSTNAME:port-distant
```

Pour expliquer je vais prendre un cas concret :

Table des matières

Je suis sur Internet avec ma machine (client), et j'ai un accès en SSH sur un serveur (srvssh). Ce serveur est dans le même réseau qu'un serveur web (srvweb avec une IP 192.169.1.50). Ce serveur Web n'est pas joignable directement depuis Internet.

On peut se connecter sur le srvssh en faisant une redirection de port 80 du srvweb sur un port de notre machine cliente :

Code BASH : **Copier vers le presse-papier**

```
ssh srvssh -L 1234:192.168.1.50:80
```

Ensuite, en saisissant sur la machine cliente dans un navigateur web http://127.0.0.1:1234 je tombe sur mon serveur web

Interdire l'ouverture d'un shell (tunnel pour redirection de port uniquement)

Si vous lancez cette commande sur un de vos serveurs pour effectuer un tunnel, il est possible d'utiliser l'option -N qui permet de ne pas ouvrir un shell. Pour empêcher avec l'utilisateur concerné d'ouvrir un shell, sur le serveur distant, il est possible après coup de modifier la ligne de la clé dans le fichier .ssh/authorized_keys en indiquant devant :

Code BASH : **Copier vers le presse-papier**

```
no-pty,no-X11-forwarding,permitopen="localhost:1234",command="/bin/echo interdit-de-lancer-un-shell" ssh-rsa......
```

Bien sûr, assurez-vous de pouvoir ouvrir un shell avec un autre utilisateur ou un autre moyen !
Il est possible par exemple, pour le tunnel, d'utiliser un utilisateur "tunnel" sur le serveur distant.

Pour plus de détails sur les tunnels, vous pouvez consulter SSH : Créer un tunnel pour rediriger des ports d'une machine à l'autre

Redirection distante de port

L'option **-R** *distante (remote)* pour rediriger un port local vers une destination (machine) distante

La syntaxe de la commande :

Table des matières

Code BASH : **Copier vers le presse-papier**

```
ssh -R port-distant:HOSTNAME:port-local  machine-distante
```

Pour expliquer je vais prendre un cas concret :

J'ai un serveur derrière un pare-feu (nommé srvssh) et j'ai un serveur à la maison (srvmaison) qui a le SSH accessible sur Internet. Je peux donc lancer depuis srvssh cette commande :

Code BASH : **Copier vers le presse-papier**

```
ssh -R 1234:localhost:22 srvmaison
```

Ainsi depuis la maison, sur le serveur concerné, si je tape

Code BASH : **Copier vers le presse-papier**

```
ssh user-srvssh@localhost -p1234
```

Le mot de passe du serveur srvssh sera demandé et on sera connecté.

Très pratique pour ouvrir un tunnel inverse 🙂

Redirection dynamique de port (Proxy SOCKS)

L'option **-D** *dynamique* pour un transfert de port dynamique basé sur SOCKS.

La syntaxe :

Code BASH : **Copier vers le presse-papier**

```
ssh -D port serveur-distant
```

Pour expliquer je vais prendre un cas concret :

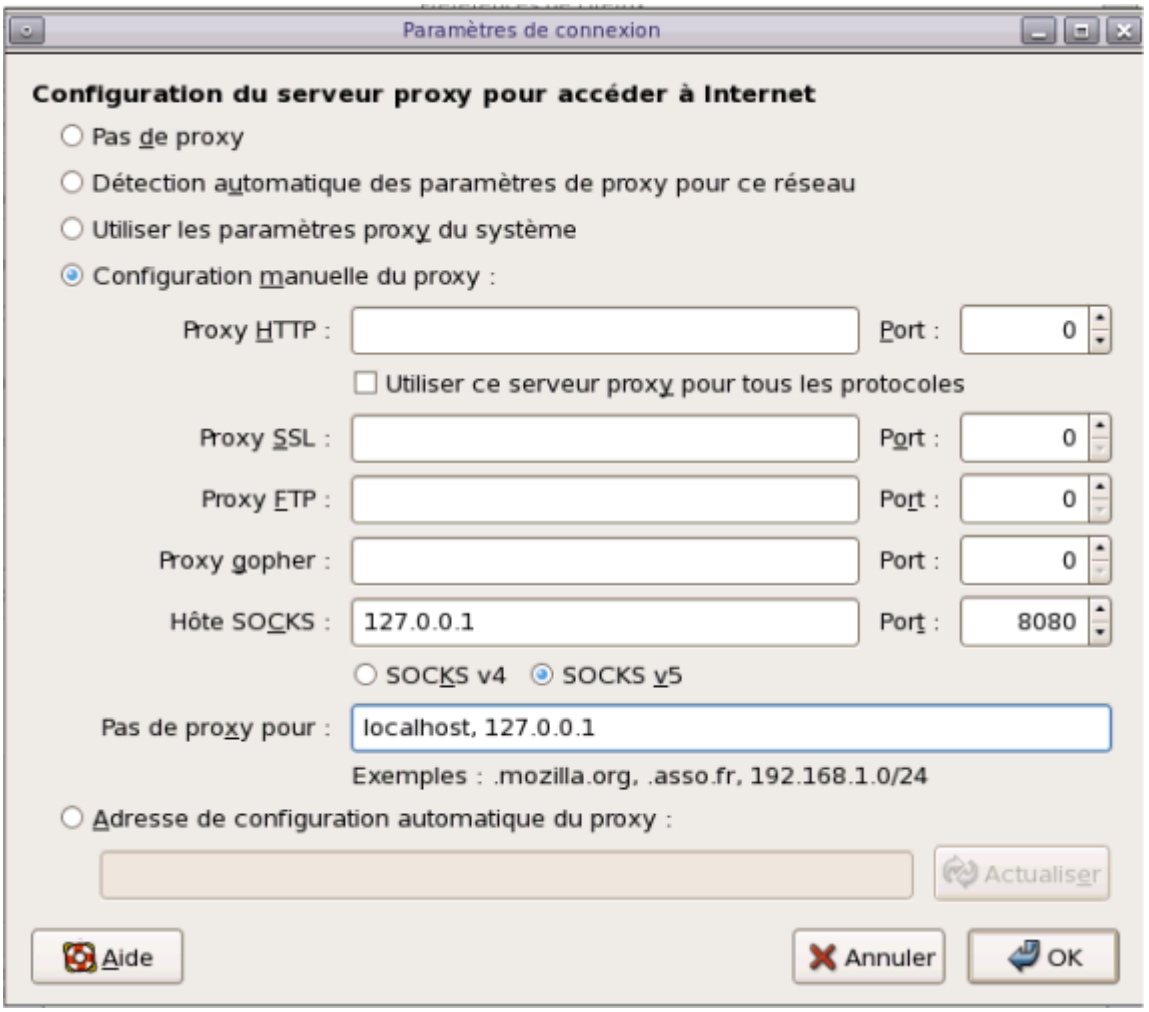
C'est assez utile si on veut naviguer via un proxy SOCKS. On est connecté sur un point d'accès Wi-Fi public avec notre PC. Tout le flux est potentiellement non sécurisé et peut être modifié.

On peut alors, si on dispose d'u Table des matières sible (chez soi par exemple) se connecter via la commande :

Code BASH : **Copier vers le presse-papier**

```
ssh -D 8080 srvmaison
```

Dans Firefox on va dans les options réseau, et on sélectionne **Configuration Manuelle du proxy** et dans **Hôte SOCKS** on renseigne **127.0.0.1** et comme **port 8080** :



Le trafic web passe par notre tunnel SSH (sécurisé depuis le PC client jusqu'à srv-maison) puis sort par le modem du réseau de srvmaison.

Cette page a été vue 263319 fois