SSH

Généralités

- SSH.

Installation

```
aptitude install ssh
yum install sshd
```

Configuration

Sur le serveur, éditer le fichier `/etc/ssh/sshd_config` et positionner les valeurs suivantes décrites dans le fichier. La configuration décrite est faite pour être sécurisée.

```
#Changer le port au lieu du 22 par défaut
Port 2222
#Utiliser uniquement la version 2 du protocole et surtout pas la 1
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key

#Après authentification création d'un autre processus SSH avec les
permissions du compte
UsePrivilegeSeparation yes

#Concerne uniquement le protocole 1
KeyRegenerationInterval 3600
#Concerne uniquement le protocole 1
ServerKeyBits 768

SyslogFacility AUTH
LogLevel INFO

#Temps maximum pour se loguer
LoginGraceTime 120
#Connexion en root impossible ni avec mot de passe ni avec clé SSH
PermitRootLogin no
#Vérification des permissions des fichiers SSH dans le dossier utilisateur
avant d'accepter le login
StrictModes yes

#Par défaut à 15. Pour enlever la déconnexion de la session trop rapide avec
```

```
le message : Write failed: Broken pipe
ClientAliveInterval 60
#ClientAliveCountMax 3
#Autoriser seulement une liste d'utilisateur séparée par des espaces
AllowUsers <user>

#Authentification complète RSA possible
RSAAuthentication yes
#Autoriser les clés publiques
PubkeyAuthentication yes
#AuthorizedKeysFile     %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for
RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

#Rendre impossible les mots de passe vide
PermitEmptyPasswords no
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes
#Par défaut à yes, les connexions par mot de passe sont désactivées. A
positionner à no uniquement lorsque les clés sont partagées et testées.
PasswordAuthentication no
# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes

#Pas de déport d'affichage pour les applications graphiques
X11Forwarding no
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no
```

```
#MaxStartups 10:30:60
#Banner /etc/issue.net
#Fichier de banner SSH qui contient le texte affiché au login
Banner /etc/ssh/banner

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes
```

Ajouter le banner SSH.

```
vi /etc/ssh/banner
```

```
**************************************************
NOTICE TO USERS WARNING! The use of this system is restricted to authorized
users, unauthorized access is forbidden and will be prosecuted by law. All
information and communications on this system are subject to review,
monitoring and recording at any time, without notice or permission. Users
should have no expectation of privacy.
**************************************************
```

```
/etc/init.d/ssh restart
```

Créer une clé publique et privée RSA

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa_<nom>
```

Entrer ou non une passphrase. Un fichier id_rsa_<nom> et id_rsa_<nom>.pub est créé dans ~/.ssh/

La cle privé reste toujours sur la machine. La clé publique doit être présente dans la configuration du serveur sur lequel on veut se connecter. De même, si on veut se connecter sur la machine qui héberge la clé privée (le serveur SSH), il faut insérer la clé publique associée dans la configuration de la machine cliente.

Propager la clé publique sur le compte utilisateur du serveur de destination sur lequel on souhaite se connecter (jamais root).

```
ssh-copy-id -i ~/.ssh/id_rsa_<nom>.pub <user>@<nom_serveur>
```

Tester la connexion en entrant la passphrase si elle est créée.

```
ssh <user>@<nom_serveur>
```

```
/etc/init.d/ssh restart
```

Sur le poste client, créer un fichier de configuration client SSH pour pouvoir utiliser plusieurs clé SSH différentes en fonction de l'hôte sur lequel on se connecte.

```
vi ~/.ssh/config
```

</code> Host @IP1

```
Port <port>
User <user1>
IdentityFile /home/<user>/.ssh/id_rsa_<nom>
```

Host toto.net

```
Port <port>
User <user2>
IdentityFile /home/<user>/.ssh/id_rsa_<nom>
```

</code>

Tester que la connexion en root ne fonctionne plus.

```
ssh root@<nom_server>
```

Lors de l'utilisation de clé SSH avec passphrase, un bug réside qui permet de se connecter tout de même sur la machine où la clé est propagée sans entrer la passphrase. Pour supprimer ce bug et obliger d'entrer la passphrase, renseigner la valeur suivante.

```
vi ~/.bash_profile
```

```
#Value to disable bug of automatic ocnnection without enter passphrase
export SSH_AUTH_SOCK=0
```

Sortir avec SSH avec le port 22 fermé

- Juste un pare-feu : facile - Pare-feu + proxy : plus embettant, il faut lui donner le nom de user et le mdp → on utilise corkscrew pour ca.

L'objectif est d'atteindre un serveur GIT qui écoute sur le port 443 à travers un proxy. Dans `/home/userdebase/.ssh/config`.

```
Host bolivar*
  Port 443
  User userdebase
  IdentityFile /home/userdebase/.ssh/id_rsa_bolivar #position de la cle
privee
  ProxyCommand /usr/bin/corkscrew 192.168.22.62 3128 %h %p
/home/userdebase/.ssh/proxy_auth
```

Ensuite faire :

```
echo 'loginproxy:mdpproxy' > /home/userdebase/.ssh/proxy_auth
chmod 600 /home/userdebase/.ssh/proxy_auth
```

Et voila on peut sortir en SSH via le port 443.

Atteindre un serveur GIT qui écoute sur le port 443 sans proxy. Dans `/home/userdebase/.ssh/config`.

```
Host bolivar*
  Port 443
  User romain
  IdentityFile /home/userdebase/.ssh/id_rsa_bolivar #position de la cle
privee
  #ProxyCommand /usr/bin/corkscrew 192.168.22.62 3128 %h %p
/home/romain/.ssh/proxy_auth
```

Si on a pas configuré le fichier config, on peut mettre les paramètres directement dans la commande : `ssh toto@boli.com -p 443`

Tunnel statique

Configurer pour écouter sur 443 et non 22 par défaut.

Clé publique du client positionnée dans `/home/toto/.ssh/authorized_keys`.

```
ssh -L port-local:localhost:port-distant nomutilisateur@nomhostdistant
```

Tunneller le port 2080 local vers le port 80 distant.

```
ssh -L 2080:localhost:80 user@srvweb.org
```

Accès transparent sur le port2.

```
nc -v localhost:2080 atteint srvweb.org sur le port 80
```

Alias pour aller plus vite

```
alias tunnelhttp='ssh -L 2080:localhost:80 user@srvweb.org'
```

Tunnel dynamique

Créez un port d'écoute qui se charge de prendre toute trame réseau qui rentre “telle quelle” et de la faire exécuter depuis l'autre bout du tunnel.

```
ssh -D port-local user@srvweb.org
ssh -D 1080 user@srvweb.org
```

Ensuite, dans un navigateur, se rendre dans éditions → préférences → avancé → réseau → paramètres de connexion. Ajouter localhost et port 1080 dans la ligne SOCKS Host

A partir de ce moment, l'ensemble du trafic du navigateur passe nativement par le tunnel sans le mentionner dans l'URL Toutes les URL entrées sont exécutées à l'autre bout du tunnel sur la machine de destination.

```
ex : https://srvbd:5432
```