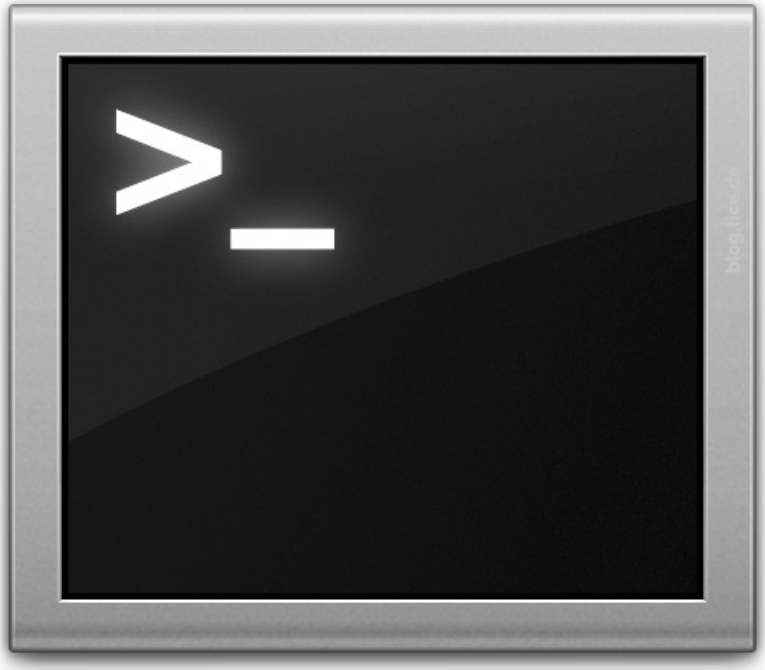
Administration Système

Summary

Administration Système



Sur ces pages je regrouperai au fur et à mesure de mes expériences professionnelles et personnelles des astuces, des commandes et tout ce qui peut être possible à qui que ce soit pour être en mesure de réaliser un minimum d'administration système.

Transfert

SCP : Transfert de fichier via SSH

SCP est un protocole de transfert de fichiers de poste à poste basé sur SSH permettant de sécuriser les échanges.

En effet, il empêche que vos informations puissent être interceptées par d'autres personnes , la sécurité et l'authentification étant gérées par SSH.

J'utilise personnellement ce protocole pour les backup des fichiers sur mon serveur. Mais il m'arrive aussi de l'utiliser pour transférer des fichiers vers mon serveur quand je n'ai pas envie d'ouvrir un logiciel tel que Filezilla. Je vais donc vous apprendre (de manière succincte) à vous servir de SCP, pour plus d'informations , taper « man scp » dans votre terminal . Notez que dans cet article les dossier sont des dossiers sur système Unix (En effet, /home/jeremy/ est mon répertoire personnel sur mon ordinateur tournant sous Debian une distribution GNU/Linux).

Backup de fichier

Serveur1 –> Serveur2 (Dans le cas d'envoi de dossier)

```
scp -r -p mylogin1@myserveur1:dossier1/ mylogin2@myserveur2:dossier2/
```

Serveur –> Ordinateur (Dans le cas d'envoi de dossier)

```
scp -r -p mylogin@myserveur:dossier/ /home/jeremy/dossier/
```

Serveur1 –> Serveur2 (Dans le cas d'envoi d'un seul fichier)

```
scp -p mylogin1@myserveur1:dossier1/mon_fichier1.txt mylogin2@myserveur2:dossier2/mon_fichier2.txt
```

Serveur –> Ordinateur (Dans le cas d'envoi d'un seul fichier)

```
scp -p mylogin@myserveur:dossier/mon_fichier.txt /home/jeremy/dossier/mon_fichier.txt
```

Envoi de fichier

Si vous désirez envoyer un dossier de votre ordinateur vers votre serveur :

```
scp -r -p /home/jeremy/dossier/ mylogin@myserveur:dossier/
```

Si vous désirez envoyer un fichier de votre ordinateur vers votre serveur :

```
scp -p /home/jeremy/dossier/ mylogin@myserveur:dossier/
```

Explications

Dans mes exemples j'ai introduit deux options en plus de la commande scp, que je vais vous expliquer directement.

L'option -r signifie « récursif », cela signifie que si vous envoyez un dossier (qui contient donc plusieurs fichiers et/ou sous-dossiers), scp parcourra tout ce dossier mais aussi les liens symboliques. Vous remarquerez que dans les commandes où je me contente de n'envoyer qu'un seul fichier, l'option -r disparaît car elle est bien entendu inutile.

L'option -p signifie que scp gardera les dates de modifications et de créations des fichiers et répertoires ainsi que leur droit en lecture et écriture.

Pour plus d'informations tournez vous vers les pages de manuel sur vos distributions en tapant tout simplement « man ssh » ou « man scp » dans un terminal. Ou alors rendez vous sur les pages wikipedia des deux protocoles SSH et SCP

Se connecter en ssh sans demande de mot de passe

J'ai une machine (A), avec un compte "benjamin" à partir de laquelle je souhaite me connecter en root sur une autre machine (B) sans demande de mot de passe.

Sur la machine A (en utilisateur benjamin) :

```
ssh-keygen -t rsa, puis 3 fois Entrée
```

Cette commande me génère une clé publique et une clé privée (dans l'ordre, id_rsa.pub et id_rsa) dans le dossier /home/benjamin/.ssh Ensuite, il faut copier la clé publique (id_rsa.pub) dans un fichier authorized_keys dans le dossier /root/.ssh de la machine B.

Sur la machine B :

```
sudo mkdir /root/.ssh
```

Créé le dossier /root/.ssh

Sur la machine A :

```
scp /home/benjamin/.ssh/id_rsa.pub 192.168.1.4:/root/.ssh/authorized_keys
```

En supposant que l'adresse IP de la machine B soit 192.168.1.4. Il faut, uniquement cette fois, taper le mot de passe root de la machine B. Cette commande copie le fichier /home/benjamin/.ssh/id_rsa.pub dans le fichier /root/.ssh/authorized_keys de la machine B (même si ce fichier n'existe pas). Toujours sur la machine A (en utilisateur benjamin) :

```
ssh root@192.168.1.4 (puis confirmer avec yes)
```

Plus aucun mot de passe ne sera demandé. Le fichier /home/benjamin/.ssh/known_hosts contient les 'identifiants' du pc sur lequel on a voulu se connecter.

Résumé :

- Je génère une clé publique ssh avec le compte benjamin sur la machine A,
- je la copie sur la machine B dans le fichier /root/.ssh/authorized_keys (si je veux avoir accès au compte root sans mot de passe),
- à partir de la machine A (avec le compte benjamin) je me connecte en ssh sur la machine B (ssh root@adresse_ip_machineB)
- il se créé donc un fichier /home/benjamin/.ssh/known_hosts contenant l'identité de la machine B.

Note : Pour améliorer la sécurité de ces connexions par clé RSA, nous pouvons restreindre l'utilisation de la clé d'authentification à l'adresse IP de la machine A en ajoutant from="" devant la clé dans le fichier authorized_keys. Ce dernier ressemblera donc à ceci :

from="192.168.1.2" ssh-rsa AB3NzaC1yc2EAzYABIwAb[...]

Note 2 : Une fois que nos clés publiques/privées ont été générées sur la machine A (étape 1), il est possible "d'**automatiser**" tout le reste avec une seule commande : **ssh-copy-id** Très simplement, sur la machine A, vous n'avez qu'à taper :

```
ssh-copy-id root@192.168.1.4
```

(pour reprendre notre exemple). Cette commande se chargera de créer le répertoire .ssh et de remplir le fichier authorized_keys automatiquement sur la machine distante.

Pour les connexions SSH sur un port autre que le port 22, il faut lancer la commande ssh-copy-id "-p 2224 root@192.168.1.4" Utilisez cette procédure avec prudence, en particulier avec les comptes root. Laisser un accès libre à une machine en root est un manque de sécurité évident, malgré toutes les précautions préalables.

ncftpget : copie récursive de fichiers depuis un serveur FTP

ncftpget est un utilitaire en ligne de commande qui permet de copier **récursivement** une arborescence de fichiers d'un serveur **FTP** distant. Quand on n'a pas d'autres moyens d'accès que le FTP c'est un petit programme bien pratique.

On peut le télécharger ici http://www.ncftp.com/ncftp/, pour Debian et Ubuntu on fera :

```
sudo aptitude install ncftp
```

Exemple d'utilisation :

Pour copier tous les fichiers du répertoire **/var/www** du serveur **www.exemple.com** avec le user FTP « **machin** » vers le répertoire **/var/www/sitemachin** de la machine locale :

```
ncftpget -R -v -u "machin" www.exemple.com /var/www/sitemachin /var/www
```

avec :

```
-R : copie récursive
-v : mode verbeux
-u "machin" : nom de l'utilisateur FTP
```

Lien : Linux: Download all file from ftp server recursively

Backup

AutoMySQLBackup

Pour sauvegarder son serveur, c'est souvent un peu chiant d'exporter les bases de données. Lorsqu'on utilise un outil comme backupPC, on est obligé de faire un peu de tuning plus ou moins propre. Heureusement qu'un outil hyper simple est disponible pour sauvegarder chaque jour, chaque semaine, et chaque mois ses bases de données MySQL : AutoMysqlBackup

Installation

Le paquet est présent dans les dépots debian/ubuntu :

```
apt-get install automysqlbackup
```

Configuration

Il n'y en a juste pas !!! Par défaut, lors de l'installation, un nouveau fichier est crée : /etc/cron.daily/automysqlbackup Il automatisera les sauvegardes quotidienne. Si vous voulez personnaliser quelques paramètres, le fichier de configuration est ici :

```
/etc/default/automysqlbackup
```

Vous pourrez régler les notifications email, le dossier de sauvegarde, les bases de données à exclure…

Enfin, le répertoire de sauvegarde est dans :

```
/var/lib/automysqlbackup/
```

Vous aurez alors 3 dossiers :

daily

weekly

monthly

Ne vous inquiétez pas si vous cherchez saturday dans daily, c'est en fait celle de weekly, pour éviter les doublons.

Les bases sont déjà compressées, et vous pouvez même les crypter.

Sécurité

Comment marche Fail2Ban ?

Développé en langage Python, **Fail2Ban** est un logiciel libre permettant d'analyser des fichiers de logs et de déclencher des actions si certaines choses suspectes sont détectées.

La grande force de Fail2Ban est sa grande modularité que cela soit au niveau des mécanismes de détections basées sur les expressions régulières ou sur les actions à mener qui peuvent aller de l'expédition d'un mail à la mise en place de règles de Firewall.

Fail2Ban se base sur un système de prisons (jails) que l'on peut définir, activer ou désactiver dans un simple fichier de configuration (/etc/fail2ban/jail.conf).

Une prison (jail) est composée, entre autres, des éléments suivants:

- Nom du fichier de log à analyser.
- Filtre à appliquer sur ce fichier de log (la liste des filtres disponibles se trouve dans le répertoire /etc/fail2ban/filter.d). Il est bien sûr possible de créer ses propres filtres.
- Paramètres permettant de définir si une action doit être déclenchée quand le filtre correspond ("match"): Nombre de "matchs" (maxretry), intervalle de temps correspondant (findtime)...
- Action à mener si nécessaire. La liste des actions se trouve dans le répertoire /etc/fail2ban/action.d. Il est également possible de créer ses propres actions.

Installation de Fail2Ban

On commence par installer Fail2Ban sur son système. Il se trouve dans la grande majorité des distributions GNU/Linux et BSD. par exemple pour l'installer sur une machine Debian 6, il suffit de saisir la commande suivante (en root ou bien précédée d'un sudo): Shell

```
apt-get install fail2ban
```

Protection contre les attaques "brute force" SSH

Un exemple étant toujours plus parlant, nous allons configurer notre Fail2Ban pour bloquer automatiquement les adresses IP des machines essayant des attaques de type "brute force" sur notre port SSH (cette attaque est à la portée de n'importe quel "neuneu". On trouve de très bon tutoriaux sur le sujet sur le net).

Si une machine cliente échoue 3 fois de suite lors de la saisie du couple login/password sur le serveur SSH alors on bloque l'accès au port TCP/SSH pendant 15 minutes. On analyse pour cela le fichier de log **/var/log/auth.log** en lui appliquant le filtre /etc/fail2ban/filter.d/sshd.conf puis, si nécessaire, l'action **/etc/fail2ban/action.d/iptables.conf**. La définition de cette prison (jail) est à faire dans le fichier **/etc/fail2ban/jail.conf**:

```
# SSH
# 3 retry ? > Ban for 15 minutes

[ssh]
enabled = true
port = ssh
filter = sshd
action = iptables[name=SSH, port=ssh, protocol=tcp]
logpath = /var/log/auth.log
maxretry = 3
bantime = 900
```

Interdire le ssh de root

Sécuriser son serveur en interdisant la connexion de l'utilisateur root

La première chose à faire est de créer un utilisateur qui pourra se connecter. Débile comme remarque ? Un âne averti en vaut deux ;)

Donc la commande de création de mon utilisateur est :

```
root@home:~# useradd -m -d /home/monutilisateurtropbien/ -s /bin/bash monutilisateurtropbien
```

Ensuite je change le mot de passe de monutilisateurtropbien :

```
root@r16721:~# passwd monutilisateurtropbien
Enter new UNIX password:
Retype new UNIX password:
passwd : le mot de passe a été mis à jour avec succès
```

VÉRIFIEZ que vous pouvez bien vous connecter en tant que **monutilisateurtropbien**.

Maintenant, il ne reste plus qu'à configurer le démon ssh pour qu'il n'accepte plus l'utilisateur root :

Interdire le login de root via ssh

Édition du fichier de configuration de sshd :

```
root@home:~# vi /etc/ssh/sshd_config
```

Remplacer :

```
PermitRootLogin yes
```

Par:

```
PermitRootLogin no
```

On redémarre le démon :

```
root@home:~# /etc/init.d/ssh restart
```

Autoriser le login de root via ssh uniquement via une clé

Vous pouvez aussi autoriser le login root uniquement à l'aide d'une clé ssh avec la directive suivante dans le fichier /etc/ssh/sshd_config :

```
PermitRootLogin without-password
```

Encore une fois pour que les modifications soient prises en compte il faut redémarrer le démon sshd :

```
root@home:~# /etc/init.d/ssh restart
```