Administration réseau sous Linux

Contenus

Articles

Administration réseau sous Linux/Configuration réseau 1
Administration réseau sous Linux/NFS 4
Administration réseau sous Linux/Samba 6
Administration réseau sous Linux/Apache 9
Administration réseau sous Linux/ProFTPD 25
Administration réseau sous Linux/DHCP 31
Administration réseau sous Linux/Netfilter 32
Administration réseau sous Linux/TCP Wrapper 38
Administration réseau sous Linux/Tcpdump 42
Administration réseau sous Linux/SSH 44
Administration réseau sous Linux/Routage 47

Références

Sources et contributeurs de l'article 51
Source des images, licences et contributeurs 52

Licence des articles

Licence 53

Administration réseau sous Linux/Configuration réseau

Installation de la carte réseau

Les cartes réseau sont souvent détectées au démarrage. Si ce n'est pas le cas il faudra charger les modules correspondants.

Pour obtenir la liste des interfaces réseau qui ont été détectées, on peut utiliser la commande

```
ifconfig -a
```

Les sections qui commencent par ethX correspondent aux cartes ethernet, où X est le numéro de la carte.

Si la carte n'est pas détectée, il faudra charger le module avec la commande

```
modprobe <nom du module>
```

Parmi les modules courants on peut noter : ne2k-pci pour les cartes NE2000, via-rhine, rtl8139...

Les modules disponibles pour votre noyau se trouvent dans `/lib/modules/<nom du noyau>/kernel/drivers/net/`. La commande suivante affiche les modules réseau disponibles pour le noyau en cours d'utilisation :

```
ls /lib/modules/`uname -r`/kernel/drivers/net/
```

Pour connaître le nom du module en fonction du nom commercial d'une carte, une recherche sur internet est souvent la meilleure solution.

Le noyau donne parfois des informations utiles sur les cartes réseau. On peut rechercher les messages contenant "eth0" pour avoir plus d'informations sur la première carte réseau détectée :

```
dmesg | grep eth0
```

La commande suivante permet d'afficher les cartes réseaux reliées au bus PCI :

```
lspci | grep Ethernet
```

Configuration de l'interface réseau

Une fois votre carte reconnue par le noyau, vous devez au moins préciser l'adresse IP et le masque de sous-réseau de la carte. Dans le cas d'un réseau local connecté à Internet, vous devez aussi ajouter l'adresse IP de la passerelle et l'adresse IP d'un ou plusieurs serveurs DNS.

Adresse IP

Pour attribuer une adresse IP à une interface réseau, on peut utiliser la commande `ifconfig` :

```
ifconfig <interface> <adresse ip>
```

Par exemple :

```
ifconfig eth0 192.168.1.12
```

Le masque de sous-réseau est déterminé automatiquement en fonction de la classe de l'adresse IP. S'il est différent on peut le spécifier avec l'option `netmask` :

```
ifconfig eth0 192.168.1.12 netmask 255.255.255.128
```

Pour voir si la carte réseau est bien configurée, on peut utiliser la commande :

```
ifconfig eth0
```

Passerelle et routage

Pour ajouter une passerelle, on peut utiliser la commande `route` :

```
route add default gw <adresse ip>
```

Pour afficher les routes vers les différents réseaux :

```
route -n
```

Tester le réseau

Pour tester si la carte réseau fonctionne, on peut essayer de communiquer avec une autre machine avec la commande

```
ping <adresse ip>
```

La commande `ping` envoi un paquet à l'adresse IP puis attend que la machine réponde. Elle affiche ensuite le temps qu'a pris toute l'opération, en millisecondes.

Informations sur les interfaces

Pour vérifier le statuts de toutes les interfaces on peut utiliser la commande

```
netstat -a
```

Nom d'hôte (*hostname*)

Le fichier **/etc/hostname** contient le nom de la machine. Il suffit de l'éditer pour changer le nom d'hôte de la machine. Cette modification n'est pas prise en compte immédiatement par le système. Elle le sera au prochain démarrage de la machine ou après avoir lancé :

```
/etc/init.d/hostname.sh
```

On peut également changer le nom d'hôte avec la commande suivante, mais il ne sera pas conservé au prochain démarrage :

```
hostname <nom d'hôte>
```

Configuration automatique au démarrage

Le fichier `/etc/network/interfaces` permet de configurer les cartes réseau de manière permanente.

Par exemple :

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
  address 192.168.1.2
  netmask 255.255.255.0
  gateway 192.168.1.1
```

Cette configuration initialisera automatiquement les interfaces "lo" et "eth0".

L'interface "lo" est souvent indispensable au système, il est important de l'initialiser. Elle aura systématiquement l'adresse IP 127.0.0.1.

L'interface "eth0" sera configurée avec l'adresse IP 192.168.1.2, le masque de sous réseau 255.255.255.0 et la passerelle 192.168.1.1 (ce paramètre est facultatif).

Si l'interface eth0 doit être configurée automatiquement par un serveur DHCP, il faut indiquer :

```
auto eth0
iface eth0 inet dhcp
```

Pour que les modifications de ce fichier soient prises en compte, il faut redémarrer ou utiliser les commandes `ifup` et `ifdown`. Par exemple :

```
ifup eth0
```

Résolution de noms d'hôte

Le fichier `/etc/host.conf` indique comment les noms doivent être résolus (c'est à dire comment passer d'une adresse IP à un nom, et inversement). Par exemple :

```
# D'abord traduire avec les serveurs DNS et ensuite avec /etc/hosts.
order bind,hosts

# Il existe des machines avec plusieurs adresses
multi on

# Vérifie l'usurpation d'adresse IP
nospoof on
```

Serveurs DNS

Le fichier `/etc/resolv.conf` contient les adresses IP des serveurs DNS. Par exemple :

```
nameserver 208.164.186.1
nameserver 208.164.186.2
search foo
```

La commande `search` indique que si un nom de domaine n'est pas trouvé, il faudra essayer en lui ajoutant `.foo`.

Fichier *hosts*

Le fichier `/etc/hosts` contient une liste de résolutions de noms (adresses IP et noms de machine). Par exemple:

```
192.168.105.2 sasa
```

Ce fichier indique que `sasa` correspond à l'adresse IP `192.168.105.2`, qui sera accessible par cet alias.

Administration réseau sous Linux/NFS

Le protocole NFS (Network file system) permet de partager des fichiers entre des machines Unix, et donc Linux. C'est un modèle client-serveur : une machine met à disposition (*exporte*) des répertoires de son système de fichier local sur le réseau. Suivant les droits d'accès, les autres stations du réseau peuvent monter ces répertoires, qui seront alors vus comme des répertoires locaux. Un ordinateur peut être à la fois client et serveur NFS.

Installation coté serveur

Commencer par vérifier que les demons NFS (nfsd) ne sont pas déjà lancés avec, par exemple, la commande

```
ps ax | grep nfsd
```

Pour lancer les démons manuellement sous Debian :

```
/etc/init.d/nfs-kernel-server start
```

ou, si c'est le serveur NFS en espace utilisateur qui est installé :

```
/etc/init.d/nfs-user-server start
```

On peut remplacer `start` par `restart` pour redémarrer le serveur.

Configuration

Pour partager (ou *exporter*) des répertoires, il faut renseigner le fichier */etc/exports*. Il indique la liste des répertoires partagés et le nom des machines qui y ont accès.

Chaque ligne correspond à un répertoire et a la forme :

```
<répertoire local> <nom ou IP des machines autorisées à se connecter>(<options>) <autres machines>(<options>)...
```

Par exemple :

```
/home/bob  ollinux(rw) station1(ro)
/projet    station1(rw) (ro)
/brouillon
```

Le serveur exporte son répertoire `/home`. La machine `ollinux` pourra le monter en lecture/écriture (`rw`), `station1` en lecture seule (`ro`), et les autres machines ne pourront pas se connecter.

De même, `station1` pourra accéder en lecture/écriture au répertoire `projet` et toutes les autres stations en lecture seule.

Enfin, tout le monde pourra accéder en lecture/écriture au répertoire `brouillon` (l'option `rw` est celle par défaut).

Pour connaitre la liste des options possibles et leur signification, consultez **man exports**.

Notez bien que les droits en écriture via le réseau seront toujours inhibés par les droits sur le système de fichier. Prenons par exemple un fichier `test` appartenant à `root`, situé dans le répertoire `projet` et avec les droits `600` (lecture/écriture pour root uniquement, aucun droit pour les autres). Si l'utilisateur `toto` accède via la station `station1` au répertoire `/projet`, il ne pourra pas accéder au fichier `test`, bien qu'il ait les "droits réseaux read-write".

Un fois le fichier `/etc/exports` correctement configuré, il suffit de relancer le service NFS par la commande suivante pour que les modifications soient prises en compte :

```
/etc/init.d/nfs-kernel-server reload
```

Installation coté client

C'est relativement simple puisque le "système de fichier réseau" NFS est directement intégré au noyau. Il suffit de vérifier que ce dernier a été compilé avec la prise en charge de NFS. C'est le cas de toutes les distributions récentes.

Pour monter un système de fichier distant, utiliser la commande `mount` avec l'option nfs :

```
mount -t nfs <machine distante>:<répertoire partagé> <répertoire local> -o <options>
```

Par exemple :

```
mount -t nfs 192.168.105.2:/armor/plages /mnt/cotes -o ro
```

Cette commande *montera* le répertoire `/armor/plages`, exporté par la station 192.168.105.2, dans le répertoire local `/mnt/cotes`, en lecture seule.

A la place d'une adresse IP, vous pouvez aussi donner un nom de machine, comme par exemple `sasa`. Pour cela, il faut que le nom `sasa` puisse être converti en adresse IP (en modifiant /etc/hosts par exemple, si on n'a pas de serveur DNS)

Connexion au démarrage

Il est possible de connecter les répertoires partagés au démarrage de la station.

Le plus simple est de renseigner le fichier `/etc/fstab` qui contient une liste des systèmes de fichiers connus.

La syntaxe est la suivante :

```
<ordinateur distant>:<répertoire distant> <répertoire local> nfs <options> 0 0
```

Pour reprendre l'exemple précédent, cela donnerait :

```
sasa:/armor/plages /mnt/cotes nfs auto,rw,user,soft 0 0
```

Les options sont décrites dans la page de man de `mount`. Certaines sont communes à d'autres systèmes de fichiers (ext2, vfat...) alors que d'autres sont spécifiques à NFS.

Administration réseau sous Linux/Samba

Samba est un service permettant de partager des répertoires et imprimantes entre des stations Linux et des stations Windows. Un How-To très complet peut être trouvé là : http:/ / www. samba. org/ samba/ docs/ man/ Samba-HOWTO-Collection/

Cette partie présente juste une introduction à l'utilisation de samba. Nous considérons que nous sommes en mode sécurité user (*security=user*, nécessite un compte utilisateur unix). L'utilisation du niveau de sécurité *domain* et la partie partie partage avec Windows (notamment les dernières versions) n'est pas vue. Pour un accès Windows, cette page donne les quelques commandes supplémentaires: http://www.oregontechsupport.com/samba/samba.php

Configuration du service Samba

Pour la configuration de ce service le principal fichier à modifier est `smb.conf` qui se trouve généralement dans `/etc` ou `/etc/samba` selon la distribution.

Il existe également des interfaces graphiques pour configurer Samba.

La section `[global]` contient les options communes à tous les répertoires partagés.

Voici quelques options utilisables :

workgroup

: Le nom du groupe de travail. Les ordinateurs du même groupe de travail se retrouvent côte à côte dans le voisinnage réseau de Windows.

server string

: La description du serveur, qui apparaitra à coté de son nom dans l'explorateur Windows. Si la description contient le terme *%h*, il sera remplacé par le nom d'hôte de la machine.

encrypt passwords

: Détermine si les mots de passe doivent être cryptés avant d'être transmis. C'est fortement recommandé et tous les systèmes Windows à partir de 98 et NT4 SP3 utilisent cette fonctionnalité par défaut.

log file

: Le nom du fichier qui contiendra le journal des activités du serveur. On peut avoir un journal par machine client en utilisant %m dans le nom du fichier. Le %m sera remplacé par le nom de la machine client.

max log size

: Taille maximale du fichier journal, en Kio.

socket options

: Indique les options à mettre sur les *sockets* comme par exemple TCP_NODELAY pour que le système envoi immédiatement les petits paquets sans attendre d'en avoir plusieurs.

De nombreuses autres options sont disponibles. Elles sont détaillées dans la page de man de `smb.conf` [1]

Exemple

```
[global]
workgroup = maison
server string = Serveur Samba sur %h
encrypt passwords = true
log file = /var/log/samba/log.%m
max log size = 1000
```

```
socket options = TCP_NODELAY
```

Configuration du partage des répertoires

Les partages Samba sont décrits dans des sections ayant la forme suivante :

```
[<nom du partage>]
<option> = <valeur>
...
```

Les paramètres principaux sont les suivantes :

comment

La description du répertoire partagé.

path

Le chemin du répertoire partagé. C'est le contenu du répertoire indiqué qui sera partagé.

read only

Détermine si les clients pourront écrire ou non dans le répertoire partagé.

public

Autoriser ou non les connexions sans mot de passe.

valid users

Liste des seuls utilisateurs autorisés à se connecter séparés par des espaces. Si on veut autoriser tous les utilisateurs il ne faut pas mettre cette option.

browseable

Détermine si le partage apparaitra dans la liste des partages du serveur.

La section `[homes]` est un partage particulier. Elle définit le partage des répertoires utilisateur des comptes unix de la machine.

De nombreuses autres options sont disponibles. Elles sont détaillées dans la page de man de `smb.conf` [1]

Par défaut (version 3.5.6) vous pouvez accéder à samba de façon anonyme (smbclient //serveur/nom_du_partage -U compte sans mot de passe). Pour effectuer un accès plus sécurisé (avec compte et mot de passe), vous devez également ajouter un compte samba qui se référence à un compte linux existant: `adduser compte (si ce n'est pas fait) smbpasswd -a compte` Les droits des répertoires et fichiers doivent être correct. Exemple `chmod u+rws,g+rx,o+rx .../dossier et/ou fichier`

Exemples

```
[cdrom]
comment = Samba server's CD-ROM
read only = yes
locking = no
path = /cdrom
guest ok = yes
```

```
[partage]
path = /media/d/partage
available = yes
browsable = yes
public = yes
```

```
writable = yes
```

```
[zelinux]
comment = Site web
path = /myrep/zelinux
read only = no
```

Protéger les répertoires partagés

Il est possible de rendre privé un répertoire et d'autoriser ou non des utilisateurs à y accéder.

Pour cela, pour chaque répertoire partagé ajoutez les options:

```
public = no
valid users = <nom des utilisateurs autorisés à accéder aux répertoires>
```

Pour chaque nom que vous avez rentré, il faut ajouter l'utilisateur samba avec

```
smbpasswd -a <nom de l'utilisateur>
```

Un compte unix du même nom doit exister. Si ce n'est pas le cas, il faut le créer avec la commande `adduser`.

Lancement du service

Lancement :

```
/etc/init.d/samba start
```

Pour le stopper :

```
/etc/init.d/samba stop
```

Pour le relancer :

```
/etc/init.d/samba restart
```

Les modifications du fichier `smb.conf` sont prises en compte pour chaque nouvelle connexion. Pour les rendre effectives sur les connexions déjà établies, il faut relancer Samba.

Accès aux répertoires

Pour accéder aux partage sous Windows, il suffit d'ouvrir le voisinage réseaux d'une station Windows et de vérifier si la machine y est.

Pour se connecter en ligne de commande à un partage à partir de Linux, on peut utiliser la commande

```
smbclient //<nom du serveur>/<nom du partage> -U <utilisateur>
```

Il est également possible de monter un partage Samba avec

```
smbmount //<nom du serveur>/<nom du partage> <répertoire local>
```

Différentes options sont disponibles. On peut les consulter dans `man smbclient` et `man smbmount`. Généralement, il est conseillé d'ajouter les options `-o username=compte,password=???` pour se connecter.

Par exemple, pour monter un répertoire "public" il faut préciser qu'il faut se connecter en *guest* :

```
smbmount //serveur/partage /point_de_montage -o guest
```

Il faut également que votre compte utilisateur ait des droits sur le montage. Le compte *root* peut utiliser *smbmount* sans trop de problème.

Autrement, plusieurs possibilités existent:

1. compléter le fichier /etc/fstab avec votre montage `//<IP-ADRESS>/<folder_on share> <your_local_mountpoint cifs defaults,iocharset=utf8,codepage=cp850,uid=1000,gid=1000,noauto,user,credentials=~/.smb 0 0)`
2. ajouter des droits dans sudoers. (par défaut sous ubuntu 10.10, `sudo smbclient //<nom du serveur>/<nom du partage> <répertoire local>-o username=compte,password=???` fonctionne bien)

Références

[1] http://us1.samba.org/samba/docs/man/manpages-3/smb.conf.5.html

Administration réseau sous Linux/Apache

Apache est un serveur HTTP libre. Un serveur HTTP permet d'héberger des sites web qui seront accessibles avec un navigateur tel que Mozilla Firefox, Internet Explorer ou encore Chrome.

Un site web peut fournir tout type de contenu (des fichiers textes, HTML, Flash, zip…). Ce contenu peut être statique (le serveur transmet un fichier au navigateur) ou dynamique (le contenu est généré par un programme exécuté par le serveur). Les sites web contiennent généralement plusieurs types de documents, certains étant statiques et d'autres dynamiques.

Nous traiterons ici d'Apache 2.2 sur un système Debian (et ses dérivés, comme Ubuntu).

Fichiers log

Par défaut sous Debian, Apache enregistre les erreurs dans le fichier `/var/log/apache2/error.log`. Quand quelque chose ne fonctionne pas, ce fichier fournit souvent des pistes pour trouver la solution.

Il enregistre également toutes les requêtes dans `/var/log/apache2/access.log`.

Configuration de base

Sous Debian, Apache se lance automatiquement lorsqu'on l'installe et à chaque démarrage du système. Lorsqu'on modifie sa configuration, il faut lui faire prendre connaissance des changements avec la commande

```
/etc/init.d/apache2 reload
```

Pour l'arrêter, le lancer ou le relancer on utilisera la même commande avec `stop`, `start` ou `restart`.

Pour d'autres systèmes il faudra consulter la documentation du système ou celle d'Apache [1].

Configuration du serveur

La configuration [2] du serveur se trouve dans `/etc/apache2/apache2.conf`. Ce fichier contient des instructions `Include` [3] qui permettent de déplacer certaines parties de la configuration dans d'autres fichiers. Debian utilise cette fonctionnalité pour les modules [4] (comme PHP) et la gestion des serveurs virtuels [5] :

Configuration des modules

Le répertoire `/etc/apache2/mods-available` contient les modules installés. Le répertoire `/etc/apache2/mods-enabled` contient les modules activés. Les modules activés sont des liens symboliques vers les modules installés.

Pour activer ou désactiver un module, on peut manipuler directement les liens ou utiliser les commandes `a2enmod` et `a2dismod (voir les pages de man)`.

Configuration des sites

De la même manière, le répertoire `/etc/apache2/sites-available` contient les sites web disponibles et `/etc/apache2/sites-enabled` les sites activés. Il en existe un préinstallé : le site `default`.

Les sites peuvent s'activer ou se désactiver en manipulant les liens dans `sites-enabled` ou en utilisant `a2ensite` et `a2dissite`.

Quelques directives classiques

La syntaxe d'Apache est assez simple. On trouve des **blocs** (ou **contextes**) comme par exemple :

```
<VirtualHost ...> # début de bloc VirtualHost
  ...
  <Directory ...> # début de bloc Directory
     ...
  </Directory>    # fin de bloc Directory
  ...
</VirtualHost>    # fin de bloc VirtualHost
```

et des **directives** comme par exemple

```
Include /etc/apache2/sites-enabled/
```

Les directives qui permettent de configurer le serveur lui-même sont généralement placées dans `apache2.conf`. Celles qui ne concernent qu'un site web sont déportées dans le fichier de configuration du site (`sites-available/mon-site-web`).

La directive DocumentRoot [6] fixe la racine du serveur Web, c'est-à-dire le répertoire de base où se trouvent les documents. Par exemple avec la directive `DocumentRoot /var/www/html`, si le navigateur demande la page `http://serveur/repertoire/fichier.txt`, le serveur cherchera le fichier `/var/www/html/repertoire/fichier.txt`.

UserDir [7] permet d'indiquer le répertoire personnel des utilisateurs du système. La directive `UserDir public_html` signifie qu'un utilisateur peut publier ses pages web personnelles dans un sous-répertoire `public_html` de son répertoire personnel. Pour l'utilisateur *toto*, c'est généralement `/home/toto/public_html`. Sa page d'accueil sera alors accessible par l'URL spéciale `http://serveur/~toto`.

DirectoryIndex [8] indique la liste des fichiers qu'Apache cherchera à afficher si l'URL n'en précise pas. Par exemple si la configuration contient `DirectoryIndex index.html index.php` et qu'on demande l'URL

`http://serveur/repertoire/`, Apache va chercher dans le répertoire un fichier `index.html` ou `index.php`. Si un de ces fichiers existe, il sera affiché. Sinon, Apache affichera soit la liste des fichiers, soit une erreur (suivant la présence de `Indexes` dans la directive Options [9]).

AccessFileName [10] définit le nom du fichier qu'on peut placer dans un répertoire pour en modifier sa configuration. Cela permet, par exemple, d'interdire localement l'affichage de la liste des fichiers, ou de protéger par mot de passe un répertoire et ses sous répertoires.

Listen [11] indique à Apache sur quel port TCP il doit écouter. Le port par défaut du protocole HTTP est 80.

ServerName [12] indique à Apache son nom de domaine et éventuellement son port. Il s'en sert lorsqu'il doit communiquer son adresse au client (le navigateur). C'est le cas par exemple lorsqu'on demande l'adresse `http://serveur/repertoire` sans *slash* (/) à la fin. Comme ce n'est pas une URL valide (l'URL d'un répertoire doit se terminer par un *slash*), Apache utilise la directive `ServerName` pour reconstruire une adresse avec un *slash* et la renvoi au client.

Gestion du nombre d'instances d'Apache

Le serveur Apache utilise plusieurs processus et prends en charge plusieurs types de stations multi-processeurs en utilisant les modules MPM (multi processing modules).

Le premier module **prefork** utilise des processus (pour systèmes stables ou plus anciens), le deuxième **worker** utilise des threads, et le dernier des threads par processus. Le dernier module **perchild** est en cours de développement et n'est pas recommandé.

Celui utilisé par défaut sous Linux est prefork.

Exemple commenté

La partie du fichier de configuration traitant la gestion du nombre de processus et la suivante:

```
##
## Server-Pool Size Regulation (MPM specific) ##

# prefork MPM
# StartServers ......... nb de processus serveur au demarrage
# MinSpareServers ...... nb minimum de processus serveurs '''libres'''
  instanciés
# MaxSpareServers ...... nb maximum de processus serveurs '''libres'''
  instanciés. S'il y en a MaxSpareServers+1 on les tues
# MaxClients ........... nb maximum de processus serveurs qui peuvent
demarrer
# MaxRequestsPerChild .. nb maximum de requètes gérées par processus
serveur.
#                        Apres MaxRequestsPerChild requètes, le
processus meurt.
#                        Si MaxRequestsPerChild=0, alors le processus
n'expire jamais.

<IfModule prefork.c>
  StartServers 5
  MinSpareServers 5
  MaxSpareServers 10
```

```
  MaxClients 20
  MaxRequestsPerChild 0
</IfModule>

# pthread MPM # StartServers ......... initial  number of server
processes to start
# MaxClients ........... maximum number of server processes allowed to
start
# MinSpareThreads ...... minimum number of worker threads which are
kept spare
# MaxSpareThreads ...... maximum  number of worker threads which are
kept spare
# ThreadsPerChild ...... constant number of worker threads in each
server process
# MaxRequestsPerChild .. maximum number of requests a server process
serves

<IfModule worker.c>
  StartServers 2
  MaxClients 150
  MinSpareThreads 25
  MaxSpareThreads 75
  ThreadsPerChild 25
  MaxRequestsPerChild 0
</IfModule>

# perchild MPM # NumServers ........... constant number of server
processes
# StartThreads ......... initial  number of worker threads in each
server process
# MinSpareThreads ...... minimum number of worker threads which are
kept spare
# MaxSpareThreads ...... maximum  number of worker threads which are
kept spare
# MaxThreadsPerChild ... maximum  number of worker threads in each
server process
# MaxRequestsPerChild .. maximum number of connections per server
process (then it dies)

<IfModule perchild.c>
  NumServers 5
  StartThreads 5
  MinSpareThreads 5
  MaxSpareThreads 10
  MaxThreadsPerChild 20
  MaxRequestsPerChild 0
  AcceptMutex fcntl
```

```
</IfModule>
```

Paramétrage des répertoires

Chaque répertoire auquel Apache accède peut être configuré indépendamment (et ses sous-répertoires en héritent).

Le paramétrage d'un répertoire se met dans un "conteneur" délimité par `<Directory chemin_du_répertoire>` et `</Directory>`. La configuration s'applique au répertoire et à tous ses sous répertoires. Si un sous-répertoire possède également sa propre configuration, elle vient s'ajouter à celle du parent.

Voici quelques exemples de contrôle d'accès. Plus de détails sont donnés dans la section "Un exemple de configuration".

```
# Configuration du répertoire racine du système
<Directory />
  # On n'autorise aucune option particulière
  Options None

  # Aucune modification n'est autorisé dans les fichiers .htaccess
  AllowOverride None
</Directory>

# Pour la racine du serveur:
<Directory /var/www/html>
  # Quelques options
  Options Indexes Includes FollowSymLinks

  # Les options peuvent être changées dans un .htaccess
  AllowOverride All

  # Permet à tout le monde d'accéder aux documents
  Allow from All

  # Spécifie comment appliquer la règle précédente
  Order allow,deny
</Directory>

# Le répertoire contenant des exécutables CGI
<Directory /usr/lib/cgi-bin>
  AllowOverride None
  Options ExecCGI
</Directory>
```

Les paramètres possibles de la directive Options [13] sont : "None", "All", "Indexes", "Includes", "FollowSymLinks", "ExecCGI", ou "MultiViews".

Gérer les pages Web personnelles

Il est possible de permettre aux utilisateurs du système de diffuser des pages personnelles sans avoir à créer un site par utilisateur. Il faut pour cela utiliser le module `userdir`.

Le répertoire contenant le site web doit être créé dans le *home* de l'utilisateur et doit être accessible en lecture pour tous. Le nom du répertoire est défini par la directive UserDir [7]. Par défaut il s'agit du répertoire `public_html`.

L'adresse pour accéder à ces sites personnels est le nom de l'utilisateur précédé d'un *tilde* (~).

Par exemple un utilisateur *toto* sur le serveur *www.iut.clermont.fr* peut créer les pages de son site dans le répertoire `/home/toto/public_html`, et on pourra y accéder avec l'adresse : http://www.iut.clermont.fr/~toto/.

Il est possible de n'autoriser que certains utilisateurs à bénéficier du `UserDir`. Par exemple pour n'autoriser que *sasa* et *toto* à avoir un site personnel :

```
UserDir disabled
UserDir enabled sasa toto
```

Pour définir les options de ces répertoires, on peut utiliser une clause `Directory` pour le répertoire `/home/*/public_html` :

```
<Directory /home/*/public_html>
  Order allow,deny
  Allow from all
</Directory>
```

La clause `UserDir public_html` ne fonctionne que pour des utilisateurs ayant un compte sur le système. L'URL http:/ / www. iut. clermont. fr/ ~toto ne fonctionne que si *toto* est un véritable utilisateur (auquel cas l'expression Unix `~toto` a un sens), pas seulement si le répertoire `/home/toto/public_html` existe.

On peut utiliser une autre forme de UserDir pour autoriser les répertoires sans forcément qu'il y ait un compte unix associé :

```
UserDir /home/*/public_html
```

Scripts CGI

Écrire un programme CGI

Le CGI (Common Gateway Interface) n'est pas un langage, c'est une norme. Un programme CGI peut être écrit en n'importe quel langage (C, Java, PHP, bash...), du moment qu'il est exécutable et qu'il respecte certaines contraintes d'entrées/sortie.

La contrainte principale concerne la sortie du programme. Si un programme CGI génère des données sur sa sortie standard, il doit les précéder d'un header http permettant de les identifier. Voici un exemple de programme CGI écrit en bash:

```
#!/bin/bash

# Header
echo "Content-type: text/html"

# Fin de l'header
echo ""
```

```
# Contenu à afficher dans le navigateur
echo "<html><body>Bonjour</body></html>"
```

Ce script génère une page HTML.

Configurer l'accès aux scripts CGI

Pour qu'Apache prenne en charge les scripts, il est nécessaire d'effectuer un minimum de paramétrage dans la configuration du site.

La déclaration `ScriptAlias /cgi-bin` *`chemin`* précise le nom du répertoire autorisé à contenir des scripts CGI. Exemple :

```
ScriptAlias /cgi-bin /var/www/cgi-bin
```

Le chemin `/cgi-bin` n'existe pas vraiment, il est dirigé vers `/var/www/cgi-bin`, et cela permet d'écrire des URL comme `http://serveur/cgi-bin/mon_script`.

La clause suivante active l'option ExecCGI dans `/var/www/cgi-bin`, ce qui autorise Apache à exécuter les scripts sur le serveur :

```
<Directory /var/www/cgi-bin>
  Options ExecCGI
</Directory>
```

Exemple : vous écrivez un script `essai.cgi`, et vous voulez que `/home/httpd/cgi-bin` contienne les scripts. Il faut donc au moins écrire:

```
ScriptAlias /cgi-bin /home/httpd/cgi-bin
<Directory /home/httpd/cgi-bin>
  Options ExecCGI
</Directory>
```

L'appel à un script essai.cgi sera effectué par l'URL: `http://serveur/cgi-bin/essai.cgi`

Le module PHP

PHP a normalement été intégré au serveur Apache sous forme d'un module chargeable situé comme tous les autres modules d'Apache dans `/usr/lib/apache2/modules`.

Les fichiers `/etc/apache2/mods-availiable/php.load` et `/etc/apache2/mods-availiable/php.conf` contiennent les directives `LoadModule` et `AddType` qui permettent à Apache d'exécuter du PHP quand on demande un fichier se terminant par `.php`. Ils doivent être liés dans `/etc/apache2/mods-enabled` pour activer PHP. On peut utiliser pour cela la commande `a2enmod`.

En marge de Apache, PHP possède lui aussi son fichier de configuration, souvent `/etc/php.ini`. Il n'est pas particulièrement conseillé d'y intervenir sauf si on sait ce que l'on fait. On peut néanmoins y observer que PHP prend bien en compte le module d'extension MySQL, contenant les fonctions d'accès au "moteur" de base de données MySQL (qui a dû être installé à part), par la présence de `extension=mysql.so`.

En cas de modification d'un fichier de configuration, comme PHP fonctionne comme module d'Apache, il faut redémarrer Apache pour qu'il réinitialise PHP par la lecture de php.ini.

```
/etc/init.d/apache2 restart
```

Protection par mot de passe

Il existe une beaucoup de solutions pour protéger un site par mot de passe.

Apache fournit une solution simple pour protéger un répertoire et ses sous-répertoires.

Il faut pour cela utiliser le fichier `.htaccess` et maintenir un fichier de mots de passe.

Configuration de l'authentification

Le fichier `.htaccess` se place dans le répertoire où il applique les règles.

On placera dans ce fichier la définition des restrictions.

Il est impératif que la modification des paramètres d'*authentification* soit autorisée dans la configuration d'Apache.

Il faut que la directive AllowOverride [14] d'un répertoire parent contienne l'option `AuthConfig`.

Les directives à placer dans le `.htaccess` sont les suivantes :

`AuthType basic`

: type d'authentification communément adopté mais peu sécurisé

`AuthName "Mon message"`

: affichera le texte comme invite dans la boîte de dialogue

`AuthUserFile /etc/apache2/my_passwd`

: indique où vont se trouver les mots de passe

`Require valid-user`

: précise qu'il faut un compte dans le fichier de mots de passe pour accéder au répertoire

On peut aussi utiliser `Require user toto sasa` pour n'autoriser que les comptes toto et sasa.

Le type d'authentification *basic* fait circuler les mots de passe en clair. Il existe d'autres types plus sécurisés comme *digest*, qu'il est recommandé de combiner à ../HTTPS/. Voir l'article sur wikipédia pour plus de détails sur le fonctionnement.

La première requête adressée à ce répertoire protégé provoquera l'affichage d'une boîte de dialogue par laquelle l'utilisateur devra s'identifier (nom et mot de passe) :

- Si le mot de passe saisi est invalide, la boite de dialogue s'affichera de nouveau.
- S'il est valide, le navigateur l'enregistre et ne le demandera plus.

Il faudra relancer le navigateur pour qu'il le demande de nouveau.

Fichier de mots de passe

Pour maintenir le fichier de mots de passe on utilisera la commande `htpasswd` (voir page de man).

Par exemple pour créer le fichier de mots de passe `/etc/apache2/default-passwd` avec comme 1er utilisateur *toto*, on utilisera la commande

```
htpasswd -c /etc/apache2/my_passwd toto
```

Pour ajouter ou modifier un utilisateur à un fichier de mots de passe existant :

```
htpasswd /etc/apache2/my_passwd sasa
```

Serveurs virtuels (*virtual hosts*)

Apache peut gérer plusieurs sites web simultanément. Ils seront tous accessibles à partir de la même adresse IP et du même port.

Pour les différencier, Apache se sert de l'adresse demandée par le navigateur.

Par exemple si site1.com et site2.com pointent sur la même adresse IP, les URL http://site1.com/ et http://site2.com/ aboutiront sur le même serveur.

Mais au moment de la requête, le navigateur précise qu'il a demandé l'adresse http://site1.com/ ou http://site2.com/.

Apache se sert de cette information pour savoir quel site afficher. On parle de *serveur virtuel* ou *virtual host*.

Pour indiquer à Apache quel site correspond à un nom de domaine, on utilise une section `<VirtualHost *>`. Sous Debian, il y a généralement un fichier par section `VirtualHost` dans le répertoire `/etc/apache2/sites-available`.

La section devra contenir une directive ServerName [12] qui indiquera le nom associé à ce *serveur virtuel*.

Elle pourra également contenir une directive ServerAlias [15] si on veut que d'autres noms aboutissent à ce site.

Par exemple :

```
<VirtualHost *>
  ServerAdmin admin@site1.com
  DocumentRoot /home/site1/racine
  ServerName site1.com
  ServerAlias www.site1.com
  AccessLog /home/site1/access.log
  ErrorLog /home/site1/error.log
  <Directory /home/site1/racine>
    AllowOverride All
  </Directory>
</VirtualHost>
```

La documentation d'Apache sur les serveurs virtuels [5] contient des informations détaillées sur le sujet.

Pour que ce serveur virtuel fonctionne, il est impératif que les noms site1.com et www.site1.com soient connus par la machine qui tente d'y accéder (celle qui lance le navigateur).

Pour cela il y a plusieurs méthodes :

- acheter le nom de domaine en question et le configurer pour qu'il pointe sur la bonne adresse IP
- utiliser un serveur DNS qui renverra la bonne IP pour ce domaine
- modifier le fichier `hosts` sur la machine cliente pour faire correspondre ce domaine à la bonne adresse IP (voir le livre Installation et configuration d'une carte réseau)

Exemples de configuration

Voici quelques exemples de configuration. L'ensemble des directives possibles peut être consulté ici : http://httpd.apache.org/docs/2.2/mod/directives.html

Pensez que les directives doivent parfois se trouver dans `apache2.conf`, parfois dans le contexte `VirtualHost` d'un site donné.

ServerType

```
ServerType standalone
```

Cette ligne indique si le serveur Apache se lance en 'autonome' (standalone) ou via `inetd` (TCP_WRAPPER). Pour la plupart des configuration, c'est en standalone. Cette directive a disparu de Apache2, qui dispose d'un autre moyen pour définir cela. Le comportement est en fait choisi d'après le MTM (Multi-processing module) choisi.

ServerRoot

```
ServerRoot /etc/apache2
```

(config serveur uniquement, pas dans un VirtualHost)

Vous indiquez ici le répertoire d'installation d'Apache. Normalement les scripts d'installation ont bien renseigné cette ligne. Vérifiez quand même.

LockFile

```
LockFile /var/run/httpd.lock
```

(config serveur uniquement, pas dans un VirtualHost)

Laissez cette ligne comme elle est, c'est à dire en commenté pour 90% des cas (# devant).

PidFile

```
PidFile /var/run/httpd.pid
```

(config serveur uniquement, pas dans un VirtualHost)

Vérifiez bien que cette ligne est décommentée. Elle indique au script de démarrage d'enregistrer le numéro de processus d'Apache pour que lors de l'arrêt du système le processus Apache soit stoppé correctement.

ScoreBoardFile

```
ScoreBoardFile /var/run/httpd.scoreboard
```

(config serveur uniquement, pas dans un VirtualHost)

Ce fichier stocke des informations pour le bon fonctionnement d'Apache.

Timeout

```
Timeout 300
```

(config serveur uniquement, pas dans un VirtualHost)

Temps en secondes avant que le serveur n'envoie ou ne reçoive un *timeout* . Quand le serveur attend une "réponse" (ex : script CGI, connexion\ldots), si au bout de ce temps, il ne reçoit pas de réponse, il va s'interrompre et prévenir l'utilisateur de l'erreur. Laissez cette valeur par défaut à moins que vous n'effectuiez des traitements dépassant cette limite. Ne pas monter trop haut cette valeur non plus car si le programme externe à "planté", ou si une erreur est survenue, vous risquez de rendre inaccessible le serveur Apache pour trop de temps (il est toujours désagréable d'attendre pour rien).

KeepAlive

```
KeepAlive on
```

Autorise ou non les connexions persistantes (plusieurs requêtes par connexions). En fait cela permet aux utilisateurs de votre serveur de lancer plusieurs requêtes à la fois, et donc d'accélérer les réponses du serveur. Laissez cette valeur par défaut la plupart du temps. Pour de petits serveurs laissez cette option sur *on* . Pour un serveur très sollicité, dès que vous vous apercevez que le système ralentit énormément ou devient indisponible assez souvent, essayez avec la valeur *off* . Mais avant, essayez de baisser la valeur de l'option suivante.

MaxKeepAliveRequests

```
MaxKeepAliveRequests 100
```

En combinaison avec l'option précédente, indique le nombre de requêtes pour une connexion. Laissez cette valeur assez haute pour de très bonnes performances. Si vous mettez 0 comme valeur, vous en autorisez en fait un nombre illimité (attention donc). Laissez la valeur par défaut là aussi.

KeepAliveTimeout

```
KeepAliveTimeout 15
```

Valeur d'attente en secondes avant la requête suivante d'un même client, sur une même connexion, avant de renvoyer un timeout. Là aussi laisser la valeur par défaut.

MinSpareServers & MaxSpareServer

```
MinSpareServers 5
MaxSpareServer 10
```

(config serveur uniquement, pas dans un VirtualHost)

Ces valeurs servent à l'auto-régulation de charge du serveur. En fait le serveur Apache contrôle lui même sa charge, suivant le nombre de clients qu'il sert et le nombre de requêtes que demandent chaque client. Il fait en sorte que tout le monde puisse être servi et ajoute tout seul un certain nombre d'instances Apaches "idle", c'est-à-dire qui ne font rien, mais sont prêtes à servir de nouveaux clients qui se connecteraient. Si ce nombre est inférieur à `MinSpareServers` il en ajoute une (ou plusieurs). Si ce nombre dépasse la valeur de `MaxSpareServer` il en arrête une (ou plusieurs). Ces valeurs par défaut conviennent à la plupart des sites.

Listen

```
Listen 3000
Listen 12.34.56.78
Listen 12.34.56.78:3000
```

Indique au serveur des ports ou des adresses IP (il y en a une par interface réseau du serveur!), ou les deux, où il doit "écouter" les demandes de connexions, EN PLUS de l'adresse et port par défaut. Voir la directive `VirtualHost` plus loin.

BindAdress

```
BindAdress *
```

Redondant avec `Listen`, cela permet de spécifier des adresses IP d'interfaces réseau, pour écouter les requêtes. Cette directive a disparu dans Apache 2.

Port

```
Port 80
```

Redondant avec `Listen`, cela permet de spécifier le port d'écoute (80 par défaut). Cette directive a disparu dans Apache 2.

LoadModule, ClearModuleList & AddModule

```
LoadModule xxxxxx.mod libexec/yyyyyy.so
ClearModuleList
AddModule zzzz.c
```

(config serveur uniquement, pas dans un VirtualHost)

Support pour les modules DSO (Dynamic Shared Object). `LoadModule` permet de charger un module. Avant Apache 2, les directives `ClearModuleList` et `AddModule` permettaient de spécifier l'ordre d'exécution des modules, à cause de problèmes de dépendances. Apache 2 peut maintenant faire cela automatiquement, car les APIs de modules leur permet de spécifier eux-mêmes leur ordre. Sous Apache 1.*, il faut cependant y prêter une grande attention, et le maintenir à jour à l'ajout de tout nouveau module.

ExtendedStatus

```
ExtendedStatus on
```

(config serveur uniquement, pas dans un VirtualHost)

Indique si le serveur doit renvoyer des informations complètes de status (*on*) ou des informations réduites (*off*). *off* par défaut. Laissez cette valeur par défaut sauf en cas de développement et de debuggage.

User & Group

```
User nobody
Group nobody
```

Une fois le serveur démarré, il serait dangereux de lui laisser les droits `root` pour répondre aux requêtes. Il est donc possible de modifier l'utiliseur et le groupe du processus pour lui donner un minimum de droits sur la machine du serveur. (En fait si quelqu'un arrive à "exploiter" votre serveur, par exemple s'il arrive a faire exécuter du code par le serveur Apache, il hérite des droits du serveur lui même. Donc si c'est `nobody` il n'a aucun droit spécifique. Si

c'est `root` ou un utilisateur réel, il aura alors des droits lui permettant d'endommager votre système.)

ServerAdmin

```
ServerAdmin root@localhost.domainname
```

Adresse e-mail de l'administrateur du site. Cette adresse est affichée par le serveur par exemple en cas d'erreur, pour que les utilisateurs puissent en avertir l'administrateur.

ServerName

```
ServerName www.domainname
```

Adresse que le serveur va renvoyer au client web. Il est préférable de mettre une adresse résolue par DNS au lieu du nom de la machine réelle, pour que les visiteurs ne voient pas le nom réel de votre machine (utile pour la sécurité aussi).

DocumentRoot

```
DocumentRoot /var/lib/apache/htdocs
```

Répertoire racine ou se trouve vos pages Web.

Directory

```
<Directory /var/lib/apache/htdocs>
  Options Indexes FollowSymlinks Multiviews
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
```

Change les paramètres du repertoire `/var/lib/apache/htdocs`. On peut placer à l'intérieur les directives suivantes :

Options

on définit les options pour ce répertoire. Les options possibles sont les suivantes :

None	Désactive toutes les options.
All	Active toutes les options SAUF Multiviews.
Indexes	Permet aux utilisateurs d'avoir des indexs généré par le serveur. C'est à dire si l'index du répertoire (`index.html` le + souvent) est manquant, cela autorise le serveur a lister le contenu du répertoire (dangereux suivant les fichiers contenu dans ce répertoire).
FollowSymLinks	Autorise a suivre les liens symboliques.
ExecCGI	Autorise à exécuter des scripts CGI à partir de ce répertoire.
Includes	Autorise des fichiers include pour le serveur.
IncludesNOEXEC	Permet mais les includes mais empêche la commande EXEC (qui permet d'executer du code).
MultiViews	Autorise les vue multiples suivant un contexte. Par exemple permet d'afficher les pages dans un language suivant la configuration du language du client.
SymLinksIfOwnerMatch	Autorise a suivre les liens seulement si l'user ID du fichier (ou répertoire) sur lequel le lien pointe est le même que celui du lien.

AllowOverride

définit comment sont gérés les fichiers .htaccess de ce répertoire :

All	Gère tout ce qui est dans .htaccess
AuthConfig	Active les directives d'autorisations AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile, AuthName, AuthType, AuthUserFile, Require, etc.
FileInfo	Active les directives contrôlant le type de document (ErrorDocument, LanguagePriority, etc.)
Limit	Active la directive d'autorisation Limit
None	Ne lit pas le fichier .htaccess et laisse les droits "Linux" de ce répertoire.
Options	Active la directive Option

Order

Donne l'ordre d'application des règles *Allow/Deny* :

deny,allow	Si le client ne correspond à aucune règle *deny* , mais correspond à une règle *allow* , alors on autorise (*allow* par défaut).
allow,deny	Si le client ne correspond à aucune règle *allow* , mais correspond à une règle *deny* , on interdit (*deny* par defaut).\hline

Allow/Deny

Nom d'hôte	Autorise/Refuse les hôtes spécifié, les adresses IP, le nom de domaine, etc...
All	Autorise/Refuse tout le monde

A vous de placer vos règles suivant le contenu de vos répertoire accessibles par le Web. Il existe les mêmes règles pour les fichiers (<Files> </Files>) et les locations (<Location> </Location>). Voir un exemple pour les fichiers (file) plus bas.

DirectoryIndex

```
DirectoryIndex index.html index.htm index.php index.php5
```

Indique le ou les fichiers à charger lorsqu'on accède à un répertoire sans préciser de fichier. Dans cet exemple, si on accède à http://example.com/repertoire/, Apache cherchera un des fichiers mentionnés (index.html, index.htm...) et s'il en trouve un il l'affichera. S'il n'en trouve pas, il affichera la liste des fichiers ou interdira l'accès (suivant la présence ou non de l'option Indexes sur le répertoire).

AccessFileName

```
AccessFileName .htaccess
```

Nom du fichier des règles d'accès pour les règles AllowOverride. Un conseil: placez comme vu précédemment une règle file du style:

```
<Files .ht*>     #pour interdire aux visiteurs de voir le contenu des Order allow,deny
                 #fichiers .ht qui contiennent les règles de
  Deny from all  #sécurité.
</Files>
```

CacheNegotiatedDocs

```
#CacheNegotiatedDocs
```

Autorise ou pas les proxies à mettre en cache les documents (pour autoriser, enlevez le commentaire # en début de ligne)

UseCanonicalName

```
UseCanonicalName On
```

Placé sur *on* , réécrit l'URL par rapport aux valeurs `Server` et `Port` spécifiées plus haut dans le fichier httpd.conf.

Sur *off* , l'URL reste celle donnée par le client.

Attention, mettez sur *on* si vous utilisez des CGI avec des variables SERVER_NAME, car si l'URL du client n'est pas la même que celle du CGI, votre script CGI ne marchera pas.

DefaultType

```
DefaultType text/plain
```

Type mime par défaut que le serveur renvoie au clients. Convient dans la plupart des cas.

HostNameLookups

```
HostNameLookups off
```

Sur *on* , le serveur le nom du client grâce à une requête DNS inverse. Sinon, il se contente de l'adresse IP, ce qui génère beaucoup moins de trafic réseau.

ErrorLog

```
ErrorLog /var/log/error_log
```

Chemin complet du fichier où les erreurs seront enregistrées.

LogLevel

```
LogLevel warn
```

Niveau d'enregistrement des erreurs avec comme valeurs possibles, par ordre décroissant d'importance, donc croissant en bavardage:

`emerg`	urgence : le serveur devient inutilisable
`alert`	une intervention est nécessaire
`crit`	erreurs critiques (accès réseau impossible par exemple)
`error`	les erreurs dans les pages, scripts
`warn`	les erreurs non bloquantes (pages mal codées, scripts comportant des erreurs non blocantes...
`notice`	événement normal mais méritant d'être remarqué
`info`	informations utiles (comme "serveur très chargé")
`debug`	Enregistre TOUT ce qui peut se passer sur le serveur

Le niveau `crit` est le minimum recommandé, et on monte généralement à `warn`.

ServerSignature

```
ServerSignature on
```

`on`	ajoute la signature (version, OS…) du serveur lorsqu'il génère des pages lui-même (index manquant, erreur de script, etc.)
`off`	ne montre que l'erreur.
`email`	ajoute un lien vers l'email définit par `ServerAdmin`

Alias

```
Alias faux_nom nom_réel
```

permet de faire des alias de répertoires (des liens en quelque sorte) (similaire à `ScriptAlias /cgi-bin chemin_complet_des_cgi`

AddType

```
AddType type extensions
```

(sous Apache2, cette directive devrait être dans un fichier `mods-availabe/nom_module.conf, au lieu de apache2.conf`)

Spécifie que des fichiers utilisant de telles *extensions* sont du *type* précisé. Cela permet de décider quoi en faire. Pour ajouter le support PHP, le fichier `mods-enabled/php5.conf` contient par exemple :

```
  AddType application/x-httpd-php .php .phtml .php3
  AddType application/x-httpd-php-source .phps
```

AddHandler

```
AddHandler cgi-script .cgi
```

Pour utiliser les scripts CGI.

Références

[1] http://httpd.apache.org/docs/2.2/invoking.html
[2] http://httpd.apache.org/docs/2.2/configuring.html
[3] http://httpd.apache.org/docs/2.2/mod/core.html#include
[4] http://httpd.apache.org/docs/2.2/dso.html
[5] http://httpd.apache.org/docs/2.2/vhosts/
[6] http://httpd.apache.org/docs/2.2/mod/core.html#documentroot
[7] http://httpd.apache.org/docs/2.2/mod/mod_userdir.html#userdir
[8] http://httpd.apache.org/docs/2.2/mod/mod_dir.html#directoryindex
[9] http://httpd.apache.org/docs/1.3/mod/core.html#options
[10] http://httpd.apache.org/docs/2.2/mod/core.html#accessfilename
[11] http://httpd.apache.org/docs/2.2/mod/mpm_common.html#listen
[12] http://httpd.apache.org/docs/2.2/mod/core.html#servername
[13] http://httpd.apache.org/docs/2.2/mod/core.html#options
[14] http://httpd.apache.org/docs/2.2/mod/core.html#allowoverride
[15] http://httpd.apache.org/docs/2.2/mod/core.html#serveralias

Administration réseau sous Linux/ProFTPD

FTP est un protocole d'échange de fichiers.

Un serveur FTP met à disposition certains répertoires du disque, et gère une authentification par mot de passe. On se connecte à ce serveur avec un client FTP.

Ce document présente la configuration d'un serveur ProFTPd [1] sous Debian. Sa gestion des droits d'accès et sa configuration sont très proches de celles d'Apache.

Installation et lancement

Sous Debian, ProFTPD est disponible dans un package et peut s'installer avec la commande

```
apt-get install proftpd
```

Il est aussi possible de le configurer poour ses propres besoins à partir des sources. Ceci permet de spécifier les modules à utiliser.

```
tar zxvf proftpd-1.x.x.tar.gz
cd proftpd-1.x.x
./configure --with-modules=mod_ratio:mod_sql
make
make installproftpd
```

Il se lance automatiquement à l'installation et au démarrage du système.

Le script qui permet de le lancer, l'arrêter ou le relancer est **`/etc/init.d/proftpd`**.

Fichier de configuration

Le principal fichier de configuration est **/etc/proftpd/proftpd.conf**.

L'ensemble des directives sont décrites sur le site web de ProFTPD [2]. Les commandes principales sont les suivantes :

```
ServerName "'''nom'''"
description = Indique le nom du serveur qui s'affichera sur les clients
```

```
AccessGrantMsg "'''message'''"
description = Message de bienvenue.
commentaires = Le message peut contenir des jokers comme %u (ici le nom de l'utilisateur)
```

```
<Limit ...>...</Limit>
description = Autorise ou refuse l'utilisation de certaines commandes du protocole FTP.
commentaires =
Par exemple la section suivante n'autorise la commande MKDIR qu'aux utilisateurs foo et bar :

<Limit MKDIR>
  Allow foo bar
  Deny All
</Limit>
```

```
ServerType ''type''
description = Détermine la manière dont le serveur reçoit les
connexions réseau.
commentaires =
Si le type est ''standalone'', un processus père sera lancé et écoutera
 sur le réseau. Si le type est ''inet'', le serveur devra être lancé
par inetd (TCP_WRAPPER).
Dans tous les cas il y aura un processus de lancé par connexion FTP.
```

```
MaxInstances 30
description = Limite le nombre de processus simultanés autorisés
```

```
User nobody
Group nobody
description = indiquent que le serveur doit s'exécuter avec les identifiants de groupe et d'utilisateur nobody
```

```
ExtendedLog /var/log/ftp.log
description = spécifie le nom de fichier log
```

```
Umask 022
description = précise les droits à '''enlever'''  aux fichiers créés
sur FTP. 022 signifie que les droits d'écriture sont enlevés au groupe
et à ''others''  pour tout nouveau fichier.
```

```
AllowOverwrite on
description = autorise un utilisateur à écraser un fichier qui lui appartient.
```

```
UseFtpUsers on
description = active l'utilisation du fichier /etc/ftpusers qui donne la liste des utilisateur n'ayant '''pas'''  accès au serveur ftp.
```

```
AllowUser ''liste-d'utilisateurs''
description = à placer dans un contexte <Limit ...>...</Limit> définit qui est autorisé à exécuter la commande du bloc ''Limit'' courant.
```

```
DenyUser ''liste-d'utilisateurs''
description = à placer dans un contexte <Limit ...>...</Limit> définit qui n'est pas autorisé à exécuter la commande du bloc Limit courant.
```

```
AllowStoreRestart
description = autorise les clients à reprendre les uploads vers le serveur.
```

```
DefaultChdir /var/ftp
description = Indique le répertoire par défaut du serveur.
commentaires = Les utilisateurs se trouvent placés dans ce répertoire lors de la connexion.
```

```
DefaultRoot /var/ftp
description = déclare ce répertoire comme la racine du système de fichiers.
```

```
UserRatio
description = permet la gestion des ratios.
commentaires =
UserRatio ''nom''  2 10 5 4096</code> indique que l'utilisateur
''nom''  a le droit de récupérer 2 fichiers sur le serveur à chaque
fois qu'il en déposera un.
On lui octroie pour commencer un crédit de 10 fichiers.
Par ailleurs, pour 1 octet déposé, il pourra recevoir 5 octets et il
détient un crédit de 4ko. À la place d'un nom on peut aussi utiliser *
qui définit des ratios par défaut.
```

```
SaveRatios on
description = sert à préciser que nous souhaitons sauvegarder les crédits de chaque utilisateur entre deux sessions.
```

```
RatioFile /ratio/RatioFile
RatioTempFile /ratio/RatioTempFile
description = indiquent les noms de fichiers permettant de sauvegarder des informations sur les ratios des utilisateurs.
```

```
FileRatioErrMsg "Vous n'avez pas suffisamment télédéchargé de fichiers"
ByteRatioErrMsg "Vous n'avez pas assez télédéchargé d'octets"
description = indiquent qu'un utilisateur a dépassé son quota par des messages.
```

```
<Directory ''répertoire''> … </Directory>
description = Cette section indique les droits sur le répertoire et sur tout ce qu'il contient.
commentaires =
Par exemple :

<Directory /var/ftp/ratio>
  <Limit ALL>
    Deny ALL
  </Limit>
  HideNoAccess on
</Directory>
Ici, nous interdisons toute opération sur le répertoire ratio grace à la section <Limit ALL>.
```

```
HideNoAccess
description = Cache tous les éléments inaccessibles aux utilisateurs.
```

```
<Anonymous ''répertoire''>...</Anonymous>
description = Configure l'accès anonyme
commentaires =
Exemple :

<Anonymous /home/ftp>
  # Après la connexion anonyme, passe sous l'utilisateur/groupe ftp.
  User ftp
  Group ftp

  # Fait correspondre le login "anonymous" au compte unix "ftp"
  UserAlias anonymous ftp

  # Autorise les comptes sans "shell", c'est souvent le cas du compte "ftp"
  RequireValidShell off

  # Interdit l'écriture partout
  <Directory *>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>

  # Autorise l'écriture dans "incoming" mais pas la lecture
  <Directory incoming>
    <Limit READ >
      DenyAll
    </Limit>
    <Limit STOR>
      AllowAll
    </Limit>
  </Directory>
</Anonymous>
```

Pour que le serveur prenne en compte le nouveau fichier de configuration, il faut recharger le démon avec: /etc/init.d/proftpd restart

Exemple de fichier proftpd.conf:

1. This is a basic ProFTPD configuration file (rename it to # 'proftpd.conf' for
2. actual use. It establishes a single server
3. It assumes that you have a user/group
4. "nobody" for normal operation.

ServerName "ProFTPd Linux Service" ServerType standalone DefaultServer on

1. Pour autoriser les clients à résumer les téléchargements, très utile.
2. Remember to set to off if you have an incoming ftp for upload.

AllowStoreRestart on

1. Port 21 is the standard FTP port.

Port 45000

1. Umask 022 is a good standard umask to prevent new dirs and files
2. from being group and world writable.

Umask 022

1. Limitation de la bande passante en lecture:

RateReadBPS 14000

1. To prevent DoS attacks, set the maximum number of child pro cesses
2. to 30. If you need to allow more than 30 concurrent connexions
3. at once, simply increase this value. Note that this ONLY works
4. in standalone mode, in inetd mode you should use an inetd server
5. that allows you to limit maximum number of pro cesses per service
6. (such as xinetd)

MaxInstances 30

1. Set the user and group that the server normally runs at. User nobody

Group nogroup

1. Nombre maximum de clients
2. MaxClients 3

1. Number of Max Clients per host
2. MaxClientsPerHost 1

1. Nombre maximums de tentatives de login

MaxLoginAttempts 3

1. Message d'accueil après une connexion réussie

AccessGrantMsg "Bienvenue %u chez moi!"

1. Pour ne pas donner d'info sur le serveur

DeferWelcome off

1. Règles pour limiter les commandes ...

<Limit MKD RNFR RNTO DELE RMD STOR CHMOD SITE CHMOD SITE XCUP WRITE XRMD PWD XPWD>

```
DenyAll
```

</Limit>

<Global>

```
DefaultRoot /var/ftp
AllowOverwrite yes
MaxClients 3
MaxClientsPerHost 1
UseFtpUsers on
AllowForeignAddress on
ServerIdent on "ProFTP DuF's Server Ready"
AccessGrantMsg "Bienvenue sur le serveur, %u"
```

</Global>

1. Serveur Virtuel pour écriture

<VirtualHost ftp.duf.com>

```
ServerName "Mon serveur FTP virtuel numero 1"
Port 46000
Maxclients 3
MaxClientsPerHost 1
DefaultRoot /var/ftp
AccessGrantMsg "Bienvenue"
```

</VirtualHost>

Client FTP

Il existe de très nombreux clients FTP. Certains sont en mode graphique, d'autres en mode texte. Les navigateurs internet permettent également de se connecter à un serveur FTP.

ftp

Le client le plus simple et le plus répandu est la commande **ftp**. Elle existe également sous Windows en ligne de commande.

Les commandes disponibles sont décrites dans la page de man. Les principales sont : help, open, ls, get, put...

Navigateur

Pour accéder à un serveur FTP à partir d'un navigateur Internet, il faut utiliser une adresse particulière. Pour une connexion anonyme, on pourra utiliser `ftp://serveur/` ou `ftp://serveur/chemin`.

Pour se connecter avec un mot de passe, on utilisera `ftp://utilisateur:mot_de_passe@serveur/`.

Références

[1] http://www.proftpd.org
[2] http://www.proftpd.org/docs/

Administration réseau sous Linux/DHCP

Le protocole DHCP (pour Dynamic Host Configuration Protocol) est un protocole réseau dont le rôle est d'assurer la configuration automatique des paramètres réseau d'une station, notamment en lui assignant automatiquement une adresse IP et un masque de sous-réseau.

Le protocole DHCP est très souvent mis en œuvre par les administrateurs de parc de stations car il offre l'énorme avantage de centraliser la configuration des stations sur une unique machine : le serveur DHCP. Le principal danger de DHCP est qu'en cas de panne du serveur DHCP, plus aucune station n'accède au réseau.

Il y a deux utilisations principales d'un serveur DHCP :

- attribuer une configuration fixe à certains postes (on les reconnaît grâce à leur adresse MAC)
- et attribuer une configuration dynamique aux postes inconnus.

On peut par exemple donner une adresse IP fixe à certains serveurs, et attribuer des adresses variables aux autres postes. Le protocole est prévu pour qu'un poste qui revient sur le réseau récupère la même adresse qu'il avait la première fois. Elle lui est réservée un certain temps (le *lease time*).

Configuration

Le fichier de configuration principal est **/etc/dhcp/dhcpd.conf**. Sa syntaxe est décrite dans **man dhcpd.conf**.

Il possède des options globales, généralement placées au début, et des sections pour chaque hôte ou réseau à configurer.

Après chaque modification de la configuration, il faut relancer le serveur :

```
/etc/init.d/isc-dhcp-server restart
```

S'il ne se relance pas, le détail de l'erreur se trouve généralement dans `/var/log/syslog`

Interfaces

Par défaut, le serveur DHCP(Dynamic host configuration protocol) est lancé sur toutes les interfaces. Dans ce cas il est impératif de configurer un réseau par interface dans dhcpd.conf.

Pour choisir les interfaces sur lesquels le serveur est lancé, il faut modifier **/etc/default/isc-dhcp-server** en indiquant par exemple

```
INTERFACES="eth1 eth2"
```

Il est obligatoire d'avoir une section "subnet" (voir ci-dessous) pour le réseau de chaque interface.

Adresse dynamique

Pour configurer une plage d'adresses à attribuer dynamiquement aux adresses MAC inconnues, on utilise une section *subnet* dans dhcpd.conf. La section suivante attribuera par exemple des adresses comprises entre 192.168.1.101 et 192.168.1.199 :

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.101 192.168.1.199;
}
```

Adresse fixe

Pour donne une adresse fixe à un poste, il faut connaître son adresse MAC et écrire une section *host*. Par exemple la section suivante attribue l'adresse *192.168.0.47* au poste *cobalt* dont l'adresse MAC est *00:13:d4:bd:b7:9a* :

```
host cobalt {
   hardware ethernet 00:13:d4:bd:b7:9a;
   fixed-address 192.168.0.47;
}
```

Options

Le serveur DHCP peut fournir d'autres informations que l'adresse IP. Ces options peuvent être définies de manière globale en les plaçant en dehors de toute section. Elles s'appliqueront alors à toutes les sections qui ne les redéfinissent pas. Si elles sont placées dans une section particulière, elles ne s'appliquent qu'à celle-ci.

L'option *domain-name-servers* permet par exemple d'indiquer au poste les adresses des serveurs DNS. L'option *routers* indique la passerelle.

Toutes les options sont décrites dans la page de man. On peut également consulter cette documentation sur internet[1].

Références

[1] http://www.linuxmanpages.com/man5/dhcpd.conf.5.php

Administration réseau sous Linux/Netfilter

Netfilter est un module qui permet de filtrer et de manipuler les paquets réseau qui passent dans le système.

Il fournit à Linux :

- des fonctions de pare-feu et notamment le contrôle des machines qui peuvent se connecter, sur quels ports, de l'extérieur vers l'intérieur, ou de l'intérieur vers l'extérieur du réseau ;
- de traduction d'adresse (NAT) pour partager une connexion internet (*masquerading*), masquer des machines du réseau local, ou rediriger des connexions ;
- et d'historisation du trafic réseau.

`iptables` est la commande qui permet de configurer Netfilter.

Fonctionnement

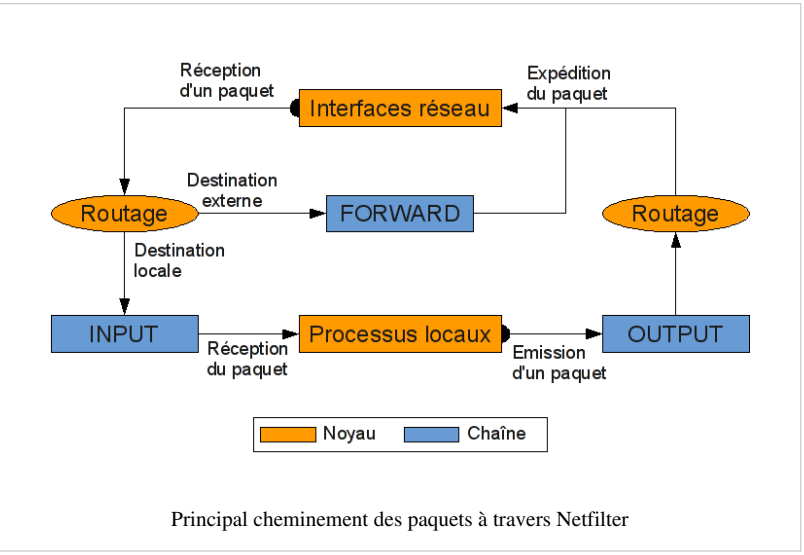
Netfilter intercepte les paquets réseau à différents endroits du système (à la réception, avant de les transmettre aux processus, avant des les envoyer à la carte réseau, etc.). Les paquets interceptés passent à travers des *chaînes* qui vont déterminer ce que le système doit faire avec le paquet. En modifiant ces chaines on va pouvoir bloquer certains paquets et en laisser passer d'autres.

Filtrage

Dans son fonctionnement le plus simple, Netfilter permet de jeter ou de laisser passer les paquets qui entrent et qui sortent.

Il fournit pour cela trois chaînes principales :

- une chaîne **INPUT** pour filtrer les paquets à destination du système,
- une chaîne **OUTPUT** pour filtrer les paquets émis par les processus du système,
- et une chaîne **FORWARD** pour filtrer les paquets que le système doit transmettre.



Principal cheminement des paquets à travers Netfilter

En ajoutant des règles dans ces chaînes on pourra laisser passer ou jeter les paquets suivant certains critères.

Chaînes

Une chaîne est un ensemble de règles qui indiquent ce qu'il faut faire des paquets qui la traversent.

Lorsqu'un paquet arrive dans une chaîne :

- Netfilter regarde la 1ère règle de la chaîne,
- puis regarde si les critères de la règle correspondent au paquet.
- Si le paquet correspond, la *cible* est exécutée (jeter le paquet, le laisser passer, etc.).
- Sinon, Netfilter prend la règle suivante et la compare de nouveau au paquet. Et ainsi de suite jusqu'à la dernière règle.
- Si aucune règle n'a interrompu le parcours de la chaîne, la politique par défaut est appliquée.

Règles

Une règles est une combinaison de critères et une cible. Lorsque tous les critères correspondent au paquet, le paquet est envoyé vers la cible.

Les critères disponibles et les actions possibles dépendent de la chaîne manipulée.

Syntaxe

La syntaxe d'`iptables` et toutes les options sont décrites dans la page de man.

Pour chaque paramètre il existe généralement une forme longue avec deux tirets (par exemple `--append`) et une forme courte avec un seul tiret (par exemple `-A`). Utiliser l'une ou l'autre n'a pas d'importance, elles sont équivalentes. Les deux possibilités sont souvent représentées dans la documentation sous la forme `--append|-A`.

Les paramètres indiqués entre crochets (par exemple `[-t <table>]`) sont facultatifs.

Ce qui se trouve entre inférieur et supérieur (par exemple `<table>`) doit être remplacé par une valeur.

La forme générale pour utiliser `iptables` est la suivante :

```
iptables [-t <table>] <commande> <options>
```

La table par défaut est la table *filter*.

Commandes

Les commandes principales sont les suivantes :

```
--list|-L [<chaîne>]
```

Affiche les règles contenues dans les chaînes ou seulement dans la chaîne sélectionnée.

Si le paramètre `-v` est placé avant cette commande, le nombre de paquets ayant traversé chaque règle sera également affiché.

```
--append|-A <chaîne> <critères> -j <cible>
```

Ajoute une règle à la fin de la chaîne <chaine>. Si tous les critères correspondent au paquet, il est envoyé à la cible. Voir plus bas pour une description des critères et des cibles possibles.

```
--insert|-I <chaîne> <critères> -j <cible>
```

Comme `--append` mais ajoute la règle au début de la chaîne.

```
--delete|-D <chaîne> <critères> -j <cible>
```

Supprime la règle correspondante de la chaîne.

```
--flush|-F [<chaîne>]
```

Efface toutes les règles de la chaîne. Si aucune chaîne n'est indiquée, toutes les chaînes de la table seront vidées.

```
--policy|-P <chaîne> <cible>
```

Détermine la cible lorsque qu'aucune règle n'a interrompu le parcours et que le paquet arrive en fin de chaîne.

Critères

Les critères possibles sont nombreux. En voici quelques uns :

```
--protocol|-p [!] <protocole>
```

Le protocole est <protocole>. Les protocoles possibles sont tcp, udp, icmp, all ou une valeur numérique. Les valeurs de /etc/protocols sont aussi utilisables. Si un point d'exclamation se trouve avant le protocole, le critère correspondra au paquet seulement s'il n'est pas du protocole spécifié.

```
--source|-s [!] <adresse>[/<masque>]
```

L'adresse source est <adresse>. Si un masque est précisé, seules les parties actives du masque seront comparées. Par exemple lorsqu'on écrit `-s 192.168.5.0/255.255.255.0`, toutes les adresses entre 192.168.5.0 et 192.168.5.255 correspondront. On peut aussi écrire le masque sous la forme d'un nombre de bits (/8 correspond à 255.0.0.0, /24 à 255.255.255.0, etc.) Le masque par défaut est /32 (/255.255.255.255), soit l'intégralité de l'adresse.

Un point d'exclamation ne fera correspondre le paquet que s'il n'a pas cette adresse source.

```
--destination|-d [!] <adresse>[/<masque>]
```

Comme --source mais pour l'adresse destination.

```
--dport [!] <port>
```

Le port destination est <port>. Il est obligatoire de préciser le protocole (`-p tcp` ou `-p udp`), car dans les autres protocoles il n'y a pas de notion de port.

```
--sport [!] <port>
```

Comme --dport mais pour le port source.

```
-i <interface>
```

L'interface réseau d'où provient le paquet. N'est utilisable que dans la chaîne INPUT.

```
-o <interface>
```

L'interface réseau de laquelle va partir le paquet. N'est utilisable que dans les chaînes OUTPUT et FORWARD.

```
--icmp-type <type>
```

Si le protocole choisi est icmp, permet de spécifier un type précis. exemples de types: *echo-request* pour l'envoi d'un "ping", *echo-reply* pour la réponse à "ping"

Cibles

Les cibles principales sont les suivantes :

```
-j ACCEPT
```

Autorise le paquet à passer et interrompt son parcours de la chaîne.

```
-j DROP
```

Jette le paquet sans prévenir l'émetteur. Le parcours de la chaîne est interrompu.

```
-j REJECT
```

Comme DROP mais prévient l'émetteur que le paquet est rejeté. La réponse envoyée à l'émetteur est également un paquet qui devra satisfaire les règles de sortie pour pouvoir passer.

```
-j LOG [--log-level <level>] [--log-prefix <prefix>]
```

Enregistre le paquet dans les logs systèmes. Au <level> par défaut, le paquet est affiché sur la console principale du système.

Cette cible est utile pour voir certains paquets qui passent (pour débugger ou pour alerter).

Utilisation simple

Le principe est assez simple à comprendre. Un paquet IP arrive sur votre machine, vous devez alors choisir ce que vous en faites. Vous pouvez l'accepter (ACCEPT), le rejeter (REJECT) ou le denier (DROP). La différence entre les deux derniers modes, est de prévenir ou non l'envoyeur, que son paquet a été refusé (avec REJECT on prévient, mais pas avec DROP).

Trois types de paquets peuvent passer par le firewall. Les paquets sortants (OUTPUT), entrant (INPUT) ou « passant », c'est-à-dire qui ne font que rebondir sur le routeur qui doit les rediriger FORWARD).

Pour organiser les règles d'acceptation/rejet, on procède de la façon suivante : – INPUT, OUTPUT, FORWARD, sont appelés des chaînes – une règle est un ensemble d'attributs auxquels correspond (ou non) un paquet : IP source, IP destination, port source, port destination, protocole . . . – quand un paquet passe par le firewall, il est aiguillé vers la chaîne correspondante – ensuite, les règles de la chaîne sont testées une par une, dans l'ordre, sur le paquet. Dès que le paquet correspond à une règle, on s'arrête. Si la règle stipule ACCEPT, le paquet est accepté. Si elle stipule DROP, il est ignoré. Si elle stipule REJECT, il est refusé avec acquittement. Les règles suivantes ne sont pas testées. – si aucune règle ne correspond au paquet, la politique par défaut est appliquée. Elle peut être positionnée à ACCEPT, DROP ou REJECT.

Il est plus sécurisant (mais plus long à mettre en place) d'utiliser une politique par défaut DROP et de créer des règles ACCEPT.

La syntaxe d'iptables est la suivante :

```
iptables -A|I chaîne -i (ou -o) interface -p protocole
--sport [port_début[:port_fin][,autre_port...]]
--dport [port_début[:port_fin][,autre_port]]
-s adresse_source -d adresse_dest -j politique
```

Il y a bien sûr de nombreuses autres options.

Créer et appliquer des règles

Toutes les commandes iptables sont tapées directement sur la ligne de commande du terminal. Il est plus pratique de les inscrire dans un fichier script et de rendre ce script exécutable (chmod +x). Ne donnez que les droits minimum à ce fichier pour qu'il ne puisse pas être lu et modifié par tout le monde. Exemple de fichier :

```
#!/bin/sh
# Effacer toutes les règles avant toute chose, afin de partir d'une
base
# propre et de savoir exactement ce que vous faites
iptables -F

# Définir une politique par défaut : le plus normal est de tout
interdire par
# défaut et de n'autoriser que certains paquets.
# "DROP" ignore les paquets, "REJECT" les refuse avec acquittement pour
 l'envoyeur
# on met souvent "DROP" pour l'INPUT (on ne donne pas d'informations à
un
# éventuel pirate) et "REJECT" pour l'OUTPUT et le FORWARD (on peut
ainsi
# récupérer des infos pour soi) mais iptables n'autorise pas REJECT
```

```
# comme politique par défaut
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Autoriser le trafic sur l'interface loopback :
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Ensuite, c'est à vous d'ajouter les règles permettant de faire
fonctionner
# les services que vous souhaitez utiliser sur votre machine.
```

Quelques exemples

Pour vider la chaine *INPUT* de la table *filter* (la table par défaut) :

```
iptables --flush INPUT
```

La politique par défaut est d'accepter tous les paquets, ce qui est généralement un mauvais choix pour la sécurité. Pour changer cette règle sur la chaine *FORWARD* de la table *filter* :

```
iptables -P FORWARD DROP
```

Pour laisser passer les paquets sur le port *telnet* qui viennent d'un réseau local (forme longue) :

```
iptables --append INPUT --protocol tcp --destination-port telnet --source
192.168.13.0/24 --jump ACCEPT
```

Pour ignorer les autres paquets entrants sur le Port (logiciel)|port *telnet* (forme courte) :

```
iptables -A INPUT -p tcp --dport telnet -j DROP
```

Pour rejeter les paquets entrants sur le port 3128, souvent utilisé par les *proxies* :

```
iptables -A INPUT -p tcp --dport 3128 -j REJECT
```

Pour autoriser telnet vers votre machine (serveur telnet) :

```
iptables -A INPUT -p tcp --dport telnet -j ACCEPT
iptables -A OUTPUT -p tcp --sport telnet -j ACCEPT
```

Pour autoriser telnet depuis votre machine (client telnet) :

```
iptables -A OUTPUT -p tcp --dport telnet -j ACCEPT
iptables -A INPUT -p tcp --sport telnet -j ACCEPT
```

Destination NAT :

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to 192.168.0.1
```

Le cas FTP

Le protocole FTP est difficile à gérer avec un firewall car il utilise plusieurs connexions. Lorsqu'on se connecte par FTP sur un serveur, on créé une connexion dite de *contrôle* qui permet d'envoyer les commandes au serveur. Ensuite pour chaque transfert de fichier et chaque listage de répertoire, une nouvelle connexion de *données* sera créée.

Le firewall peut très bien gérer la connexion de contrôle, mais pas celles de transfert car elles sont faites sur des ports indéterminés. Sur certains serveurs FTP il est possible de fixer une plage de ports à utiliser, et dans ce cas on peut faire du NAT simple.

Il est également possible d'utiliser le "conntrack ftp". C'est un module pour Netfilter qui inspecte les connexions FTP de contrôle pour détecter les connexions de données. Il indique alors au noyau que ces connexions sont liées à une autre connexion (*RELATED*). Pour autoriser ces connexions avec iptables on utilise le module *state*.

Pour cela il faut charger le module ip_conntrack_ftp :

```
modprobe ip_conntrack_ftp
```

Et autoriser les connexions *RELATED* en entrée et en sortie :

```
iptables -A INPUT -m state --state RELATED -j ACCEPT
iptables -A OUTPUT -m state --state RELATED -j ACCEPT
```

Administration réseau sous Linux/TCP Wrapper

Le super service inetd permet de contrôler et de restreindre l'accès à certains services réseau. Ceux-ci sont gérés par inetd et ne sont plus en mode dit "standalone".

Inetd utilise le démon tcpd qui intercepte les demandes de connexion à un service et vérifie par le biais des fichiers hosts.allow et hosts.deny si le client est autorisé à utiliser ce service. Sur les versions de linux actuelles, il est installé par défaut. Par contre il n'est pas actif dans sa partie contrôle d'accès.

Inetd est un élément à mettre en œuvre pour sécuriser une machine sous linux, il ne peut toutefois pas remplacer complétement un vrai firewall.

Le principe

Lorsque vous souhaitez vous connecter sur une machine distante avec la commande telnet par exemple, le démon *inetd* intercepte votre demande de connexion et vérifie dans le fichier *inetd.conf* si le service telnet est utilisable. Si la réponse est positive, votre demande est passée à *tcpd* qui vérifie dans les fichiers *hosts.allow* et *hosts.deny* si vous avez le droit de vous connecter en telnet, si cela est le cas votre demande de connexion sera autorisée, sinon vous serez rejeté. Dans tous les cas de figure et cela est l'autre fonction de TCP_wrappers, *tcpd* transmettra à *syslogd* (démon de log) votre demande (cette demande se retrouvera dans les logs, fichier /var/log/securite).

L'installation

Par défaut il est installé avec la plupart des distributions, mais au cas où, le paquet à installer est : tcp_wrappers-x.....rpm. TCP_wrappers utilise les fichiers suivants : tcpd, inetd, inetd.conf, hosts.allow, hosts.deny, tcpdchk, tcpdmatch.

Configuration : le fichier inetd.conf

Ce fichier se trouve dans le répertoire */etc*. Vous pouvez activer ou désactiver ici des services, en plaçant un # devant la ligne ou en l'enlevant, puis en obligeant la relecture du fichier avec la commande *killall -HUP inetd*. Il est possible d'ajouter d'autres services dans ce fichier.

En voici un commenté:

```
# Version: \@(#)/etc/inetd.conf
# Les premières lignes sont utilisées par  inetd
#
#echo stream tcp nowait root internal
#echo     dgram     udp     wait     root internal
#discard     stream     tcp     nowait     root internal
#discard     dgram     udp     wait     root internal
#daytime     stream     tcp     nowait     root internal
#daytime     dgram     udp     wait     root     internal
#chargen     stream     tcp     nowait     root     internal
#chargen     dgram     udp     wait     root     internal
#time     stream     tcp     nowait     root     internal
#time     dgram     udp     wait     root     internal
#
#ftp et telnet sont deux services très utilisés.
#Ils ne sont pas spécialement sécurisés. telnet peut être remplacé  par ssh qui est beaucoup plus sécurisé.
#
# ftp     stream     tcp nowait     root     /usr/sbin/tcpd
in.ftpd telnet stream tcp     nowait     root     /usr/sbin/tcpd
in.telnetd
#
#Shell, login, exec, comsat et talk sont des protocoles BSD.
#Essayez de ne pas les utiliser.
#Ils présentent des failles au niveau de la sécurité.
# shell     stream tcp nowait     root /usr/sbin/tcpd
in.rshd login stream tcp nowait root /usr/sbin/tcpd in.rlogind
#exec stream     tcp nowait     root /usr/sbin/tcpd in.rexecd
#comsat dgram udp wait     root /usr/sbin/tcpd in.comsat
talk gram udp     wait nobody.tty /usr/sbin/tcpd in.talkd
ntalk dgram     udp wait     nobody.tty /usr/sbin/tcpd in.ntalkd
#dtalk stream tcp wait nobody.tty /usr/sbin/tcpd in.dtalkd
#
# pop3 et imap sont les serveurs de messagerie.
# A n'activer que si vous les utilisez.
# Oubliez pop2
#pop-2 stream     tcp nowait root
```

```
/usr/sbin/tcpd ipop2d
#pop-3 stream     tcp nowait root
/usr/sbin/tcpd     ipop3d
#imap stream     tcp nowait root
/usr/sbin/tcpd     imapd
#
#Le service UUCP est un moyen d'envoyer des fichiers entre machines.
#Ce service n'est pratiquement plus utilisé.
#Evitez de l'utiliser.
#uucp stream     tcp     nowait     uucp
/usr/sbin/tcpd /usr/lib/uucp/uucico -l
#ftp et bootp sont utilisés pour permettre aux machines
 clientes qui ne disposent pas
# de disque de booter, de recevoir une adresse IP, de charger le système.
#TFTP ne disposant pas de système d'authentification il
#est un énorme trou de sécurité. # Vous devez absolument éviter de
 l'utiliser
# tftp     dgram udp     wait     root
/usr/sbin/tcpd in.tftpd
# bootps     dgram     udp     wait root /usr/sbin/tcpd bootpd
#Finger, cfinger, systat and netstat ne sont pas
# dangereux en eux même, mais ils
# fournissent des informations sur les comptes
#utilisateurs et votre système.
#Il ne faut donc pas les utiliser.
# finger stream tcp nowait nobody /usr/sbin/tcpd in.fingerd
#cfinger stream tcp nowait root /usr/sbin/tcpd in.cfingerd
#systat stream tcp nowait guest /usr/sbin/tcpd /bin/ps -auwwx
#netstat stream tcp nowait guest /usr/sbin/tcpd     /bin/netstat -f inet

# Service d'authentification auth fournit des information sur l'utilisateur
auth stream tcp wait root /usr/sbin/in.identd in.identd -e -o
# end of inetd.conf linuxconf stream tcp wait root /bin/linuxconf linuxconf --http
```

Le # devant une ligne rend la ligne inactive, donc le service non disponible. Si vous ne devez pas utiliser un service, rendez le inactif.

Voici un descriptif de quelques options. Considérons la ligne suivante:

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd
```

- **ftp:** nom du service, tel qu'il est déclaré dans /etc/services
- **stream:** type de service de transport de données (il existe stream pour tcp, dgram pour udp, raw pour IP)
- **tcp:** nom du protocole tel qu'il existe dans /etc/protocols
- **wait:** état d'attente, si l'état est wait inetd doit attendre que le serveur ait restitué la socket avant de se remettre à l'écoute. On utilise wait plutôt avec les types dgram et raw. l'autre possibilité est nowait qui permet d'allouer dynamiquement des sockets à utiliser avec le type stream.
- **root:** Nom de l'utilisateur sous lequel le daemon tourne

- **/usr/sbin/tcpd in.ftpd** Chemin d'accès au programme in.ftpd lancé par inetd (il est possible ici d'ajouter les options de démarrage du programme.

Hosts.allow et Hosts.deny

Vous trouverez ces deux fichiers dans le répertoire /etc.

Le premier fichier lu est *hosts.allow* , puis *hosts.deny*. Si une requête est autorisée dans le fichier hosts.allow alors elle est acceptée, quelque soit le contenu du fichier hosts.deny. Si une requête ne satisfait aucune règle, que ce soit dans hosts.allow ou hosts.deny alors elle est autorisée. En un mot si vous ne mettez rien dans hosts.deny, alors vous n'avez rien fait.

Voici un petit exemple qui dans des cas simples est suffisant :

fichier **hosts.allow**

```
# hosts.allow
ALL : LOCAL
in.ftpd : 192.168.0.,10.194.168.0/255.255.255.0, 192.168.1.1
in.telnetd : .iut.u-clermont1.fr
```

On autorise tout les ports depuis un accès local, et on autorise ftp pour les machines venant du réseau 192.168.0.0, ainsi que les machines du réseau 10.194.168.0 avec une autre notation et enfin la seule machine qui a pour adresse 192.168.1.1

fichier **hosts.deny**

```
#hosts.deny
ALL:ALL
```

Le fichier hosts.deny est simple à comprendre, il interdit tout par défaut. Le fichier hosts.allow indique les services que je veux autoriser (Le nom du service doit être identique au nom qui se trouve dans inetd.conf). la syntaxe est la suivante :

```
deamon[,deamon,...] : client[,client,...] [ : option : option ...]
```

Cette syntaxe est identique dans les deux fichiers, hosts.allow et hosts.deny.

Les utilitaires de tcp wrappers

- tcpdchk -av : permet de voir la configuration de tcp wrappers
- tcpdmatch in.ftpd localhost pour simuler une connexion sur in.ftpd

voir aussi xinetd un équivalent de tcp_wrappers, mais avec plus d'options.

Administration réseau sous Linux/Tcpdump

Dans un réseau ethernet relié par un concentrateur (ou *hub*), chaque machines reçoit tous les paquets qui circulent sur le réseau. En fonctionnement normal, les cartes réseau ne réceptionnent que les paquets qui leur sont destinés, mais on peut faire en sorte qu'elles transmettent tous les paquets au système.

Les *hub* sont de moins en moins utilisés. Ils sont généralement remplacés par des commutateurs (ou *switch*) qui savent déterminer (en fonction de l'adresses MAC) sur quel câble il faut envoyer un paquet. Les machines ne reçoivent donc généralement que les paquets qui leur sont destinés.

L'utilitaire tcpdump permet d'inspecter les paquets qui sont reçus et transmis par une carte réseau.

Filtrage

Il est possible de sélectionner les paquets à "écouter" en fonction d'expressions. Ainsi, ne seront affichées / traitées que les informations pour lesquelles le résultat de l'expression est vérifié. Une expression est composée de primitives et d'opérateurs logiques.

Une primitive est un identifiant précédé de mots clés qui indiquent le type de l'identifiant. Par exemple la primitive *src port 21* contient les éléments suivants :

- le mot clé *src* qui indique que l'identifiant ne porte que sur la source du paquet
- le mot clé *port* qui indique que l'identifiant est le port du paquet
- l'identifiant 21

La primitive correspond donc au *port source 21*.

De la même manière, la primitive *ether src 00:11:22:33:44:55* indique *l'adresse ethernet (ou MAC) source 00:11:22:33:44:55*.

Les primitives les plus courantes sont les suivantes :

src <adresse>

: l'adresse source est <adresse>

dst <adresse>

: l'adresse destination est <adresse>

host <adresse>

: l'adresse source ou destination est <adresse>

port <port>

: le port source ou destination est <port>

src port <port>

: le port source est <port>

dst port <port>

: le port destination est <port>

portrange <port1>-<port2>

: le port est compris entre <port1> et <port2>. On peut préciser l'origine avec les mots clés *src* ou *dst* et le protocole avec les mots clés *tcp* ou *udp*.

Les primitives peuvent être reliées avec les opérateurs logiques *and*, *or* et *not*. Par exemple l'expression suivante va trouver tous les paquets en provenance de tiny mais dont le port n'est pas le port ssh :

```
src tiny and not port ssh
```

Il est également possible de spécifier un protocole: udp, tcp, icmp.

Options

Plusieurs options permettent de modifier le comportement de tcpdump :

-i <interface>

sélectionne l'interface réseau sur laquelle tcpdump écoute. Par défaut il prend la première active (sauf *lo*).

-x

affiche également les données contenues dans les paquets trouvés, sous forme hexadécimale

-X

affiche les données des paquets sous forme ASCII

-s <nombre>

par défaut, seuls les 68 premiers octets de données sont affichés. Ce paramètre permet de modifier ce nombre.

-w

pour spécifier le chemin d'un fichier où sauvegarder le dump.

Exemples

```
tcpdump src 192.168.0.1
```

Ici, les seuls paquets affichés sont ceux en provenance de 192.168.0.1. Nous pouvons également préciser nos préférences en ajoutant un critère :

```
tcpdump src 192.168.0.1 and port 80
```

Là, le seul port qui nous intéresse est 80 (http).

Voici une ligne complète qui ne laisse vraiment passer que les paquets en provenance de 192.168.0.1 vers 212.208.225.1, sur le port 53 en udp.

```
tcpdump -x -X -s 0 src 192.168.0.1 and dst 212.208.225.1 and port 53 and udp
```

Nous avons demandé l'affichage du contenu des paquets au format hexadécimal et ascii (-x -X) et ce, quelle que soit leur taille (-s 0). Nous obtenons les informations désirées :

```
0x0000:  4500 003b 0000 4000 4011 ca00 c1fd d9b4  E..;..@.@.......
0x0010:  c1fc 1303 80a1 0035 0027 213d 14c2 0100  .......5.'!=....
0x0020:  0001 0000 0000 0000 0377 7777 056c 696e  .........www.lin
0x0030:  7578 036f 7267 0000 0100 01              ux.org.....
```

Administration réseau sous Linux/SSH

SSH signifie Secure SHell. C'est un protocole qui permet de faire des connexions sécurisées (i.e. cryptées) entre un serveur et un client SSH.

On peut l'utiliser pour se connecter à une machine distante comme avec telnet, pour transférer des fichiers de manière sécurisée ou pour créer des tunnels. Les tunnels permettent sécuriser des protocoles qui ne le sont pas en faisant passer les données par une connexion SSH.

Le système de clés de SSH

Cryptographie asymétrique

SSH utilise la cryptographie asymétrique RSA ou DSA. En cryptographie asymétrique, chaque personne dispose d'un couple de clé : une clé publique et une clé privée. La clé publique peut être librement publiée tandis que chacun doit garder sa clé privée secrète. La connaissance de la clé publique ne permet pas d'en déduire la clé privée.

Si la personne A veut envoyer un message confidentiel à la personne B, A crypte le message avec la clé publique de B et l'envoie à B sur un canal qui n'est pas forcément sécurisé. Seul B pourra décrypter le message en utilisant sa clé privée.

Cryptographie symétrique

SSH utilise également la cryptographie symétrique. Son principe est simple : si A veut envoyer un message confidentiel à B, A et B doivent d'abord posséder une même clé secrète. A crypte le message avec la clé sécrète et l'envoie à B sur un canal qui n'est pas forcément sécurisé. B décrypte le message grâce à la clé secrète.

Toute autre personne en possession de la clé secrète peut décrypter le message.

La cryptographie symétrique est beaucoup moins gourmande en ressources processeur que la cryptographie asymétrique, mais le gros problème est l'échange de la clé secrète entre A et B. Dans le protocole SSL, qui est utilisé par les navigateurs Web et par SSH, la cryptographique asymétrique est utilisée au début de la communication pour que A et B puissent s'échanger une clé secrète de manière sécurisée, puis la suite la communication est sécurisée grâce à la cryptographie symétrique en utilisant la clé secrète échangée.

Configuration du serveur SSH

Pour manipuler le daemon (le lancer, l'arrêter, recharger la configuration...), on utilise la commande

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

Le fichier de configuration du serveur SSH est /etc/ssh/sshd_config. À ne pas confondre avec /etc/ssh/ssh_config qui est le fichier de configuration du client SSH.

Parmi les multiples options, on peut noter :

- **Port 22:** Signifie que le serveur SSH écoute sur le port 22, qui est le port normal de SSH. Il est possible de le faire écouter sur un autre port en changeant cette ligne.
- **Protocol 2:** Signifie que le serveur SSH accepte uniquement la version 2 du protocole SSH. C'est une version plus sécurisée que la version 1 du protocole. Pour accepter les deux protocoles, change la ligne en : Protocol 2,1
- **PermitRootLogin no:** Signifie que l'on ne peux pas se logger en root par SSH. Pour logger en root, il suffit de se logger en utilisateur normal et d'utiliser la commande su.
- **X11Forwarding yes:** Autorise le transfert par SSH de l'affichage graphique.
- **LoginGraceTime 600:** Temps de connexion maximum

- **RSAAuthentication yes:** Méthode d'authentification.
- **AuthorizedKeysFile .ssh/authorized_keys** fichier utilisé pour 'l'autologin'
- **PermitEmptyPasswords no:** permet ou non les mots de passe vide

Si le fichier de configuration du serveur a été modifié, il faut indiquer au démon sshd de relire son fichier de configuration, avec la commande */etc/init.d/ssh restart*.

Se logguer par SSH

Deux types d'authentification sont possibles : par mot de passe et par clé. Dans les deux cas on utilise une des commandes suivantes :

```
ssh -l <login> <adresse du serveur SSH>
ssh <login>@<adresse du serveur SSH>
```

Authentification par mot de passe

C'est la méthode la plus simple. Lors de la connexion, le client ssh demande le mot de passe du compte. Dans ce cas, ssh crypte le mot de passe ce qui évite de le voir circuler en clair sur le réseau.

Authentification par clé

Au lieu de s'authentifier par mot de passe, les utilisateurs peuvent s'authentifier grâce à la cryptographie asymétrique et son couple de clés privée/publique, comme le fait le serveur SSH auprès du client SSH. La clé publique est placée sur le serveur dans le home du compte sur lequel on souhaite se connecter. La clé privée reste sur le poste client. Dans ce cas, aucun mot de passe ne circule sur le réseau.

Générer une clé

Pour générer un couple de clés, on utilise la commande :

```
ssh-keygen -t dsa
```

Deux clés vont être générées, une clé publique (par défaut `~/.ssh/id_dsa.pub`) et une clé privée (par défaut `~/.ssh/id_dsa`). C'est la clé publique qu'il faudra copier sur le serveur.

Les clés générées ont par défaut une longueur de 1024 bits, ce qui est aujourd'hui considéré comme suffisant pour une bonne protection.

La commande demande un nom de fichier pour sauvegarder la clé privée et un nom de fichier pour sauvegarder la clé publique. Par défaut, la clé privée est stockée dans le fichier $HOME/.ssh/id_dsa.

La clé privée est enregistrée avec les permissions 600. La clé publique porte le même nom de fichier suivi de ".pub", avec les permissions 644.

Lors de la création de la clé, l'utilitaire demande une *pass phrase* qui est un mot de passe pour protéger la clé privée (2eme protection). Cette pass phrase sert à crypter la clé privée. La pass phrase sera alors demandée à chaque utilisation de la clé privée, c'est à dire à chaque fois que vous vous connecterez en utilisant cette méthode d'authentification. Un mécanisme appelé ssh-agent permet de ne pas rentrer le mot de passe à chaque fois (voir les docs).

Il est possible de changer la pass phrase qui protège la clé privée avec la commande *ssh-keygen -p*.

Autoriser sa clé publique

Pour autoriser une clé à se connecter à un compte, il faut placer sa partie publique dans le fichier *$HOME/.ssh/authorized_keys* du compte en question, sur le serveur SSH. Si vous souhaitez vous connecter au serveur sur le compte sasa, le fichier est /home/sasa/.ssh/authorized_keys.

Pour transférer la clé publique, on peut utiliser ftp, scp (copie de fichier par ssh), ou un simple copier/coller entre deux terminaux (c'est simplement une longue ligne de caractères ASCII).

Chaque ligne du fichier authorized_keys correspond à une clé publique autorisée à se connecter. Il faut vérifier que chaque clé est bien sur une seule ligne, sinon ça ne fonctionne pas.

Le répertoire *$HOME/.ssh' devrait être protégé en écriture, avec les permissions 755 (ou 705).* De la même manière, le fichier authorized_keys ne doit pas être lisible par tous (600 par exemple).

Ensuite, pour se logger, il suffit de procéder comme précédemment.

Si la clé privée a été enregistré dans un autre fichier que $HOME/.ssh/id_dsa, il faut le préciser au client ssh :

```
ssh –i <nom du fichier contenant la clé privée> <login>@<serveur>
```

Agent SSH

Un agent SSH est un programme qui garde en mémoire des clés privées. Le principe est le suivant :

- on lance un agent
- on lui ajoute des clés (si les clés sont cryptées, elles sont décryptées avec la passphrase avant d'être ajoutées)
- à chaque connexion ssh, les clé de l'agent sont utilisées en priorité

Les avantages principaux sont :

- que la passphrase n'est demandée qu'une seule fois au moment où elle est ajoutée à l'agent,
- et que l'agent est capable de faire suivre la clé sur plusieurs connexions.

Pour lancer un agent on utilise généralement une commande qui ressemble à :

```
ssh-agent /bin/bash
```

L'agent SSH lance un nouveau shell ("/bin/bash") dans lequel il sera actif. Il ne sera utilisable qu'à partir de ce shell et dans les programmes qui y seront lancés.

Pour ajouter une clé on utilise

```
ssh-add [<fichier>]
```

Si on ne précise aucun fichier, il utilisera la clé par défaut ("~/.ssh/id_dsa" pour SSH 2).

Si la clé est crypté, la passphrase sera demandée et la clé décryptée sera ajoutée à l'agent.

Toutes les connexions SSH (avec ssh, scp...) lancées à partir de ce shell utiliseront l'agent et ne demanderont donc plus la passphrase.

Créer un "tunnel" crypté entre deux stations

SSH est également capable de fournir un encryptage à d'autres services (ftp par exemple) par le biais du *port forwarding*.

(options -L et -R de la commande ssh), de la manière suivante:

Considérons deux stations HOST1 et HOST2. Supposons que sur la machine HOST1, vous utilisiez la commande :

```
ssh –L p1:HOST2:p2 HOST2
```

ou sur HOST2:

```
ssh –R p1:HOST2:p2 HOST1
```

alors vous obtenez un tunnel sécurisé dans lequel vous pouvez passer toute connexion, lesquelles seront automatiquement cryptées.

Sur HOST1, *ssh -L p1:HOST2:p2 HOST2* signifie, que lorsqu'on se connecte au port p1, les paquets sont retransmis vers le port p2 de la machine HOST2 via HOST2.

Administration réseau sous Linux/Routage

Adresses IP et MAC

Chaque interface de chaque ordinateur sera identifié par

- Son adresse IP : une adresse IP (version 4, protocole IP V 4) permet d'identifier un hôte et un sous-réseau. L'adresse IP est codée sur 4 octets - 32 bits. (les adresses IP V 6, ou IP next generation seront codées sur 16 octets - 128 bits).
- L'adresse mac de sa carte réseau (carte ethernet ou carte wifi) sur 6 octets - 48 bits;

Une adresse IP permet d'identifier un hôte. Une passerelle est un ordinateur qui possède plusieurs interfaces et qui transmet les paquets d'une interface à l'autre. La passerelle peut ainsi faire communiquer différents réseaux. Chaque carte réseau possède une adresse MAC unique garantie par le constructeur. Lorsqu'un ordinateur a plusieurs interfaces, chacune possède sa propre adresse MAC et son adresse IP. On peut voir sa configuration réseau par ifconfig :

```
$ ifconfig eth0
eth0        Link encap:Ethernet HWaddr 00:B2:3A:24:F3:C4
            inet addr:192.168.0.2 Bcast:192.168.0.255 Mask:255.255.255.0
            inet6 addr: fe80::2c0:9fff:fef9:95b0/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:6 errors:0 dropped:0 overruns:0 frame:0
            TX packets:16 errors:0 dropped:0 overruns:0 carrier:5
            collisions:0 txqueuelen:1000
            RX bytes:1520 (1.4 KiB) TX bytes:2024 (1.9 KiB)
            Interrupt:10
```

On voit l'adresse MAC 00:B2:3A:24:F3:C4 et l'adresse IP 192.168.0.2. Cela signifie que le premier octet de l'adresse IP est égal à 192, le deuxième 168, le troisième octet est nul, et le quatrième vaut 2.

Sous-réseaux

Classes de sous-réseaux

Jusqu'au années 1990, les adresses IPv4 étaient organisées en sous-réseaux. Chaque sous-réseau possède une adresse, qui est une partie de l'adresse IP des machines de ce sous-réseau.

Par exemple, l'adresse IP 192.168.4.35 appartient au sous-réseau 192.168.4, parfois aussi noté 192.168.4.0.

Les sous-réseaux sont organisés en classes. Chaque classe de sous-réseaux correspond à des réseaux pouvant contenir un certain nombre de machines.

- Classe A : les adresses de de 1.0.0.0 à 127.0.0.0. L'identifiant du réseau est alors sur 8 bits et les identifiants de machine sont sur 24 bits (plusieurs millions de machines par sous-réseau ;
- Classe B : les adresses de de 128.0.0.0 à 191.255.0.0. L'identifiant du réseau est alors sur 16 bits et les identifiants de machine sont sur 16 bits (plus de 65 000 machines par sous-réseau) ;
- Classe C : les adresses de de 192.0.0.0 à 223.255.255.0. L'identifiant du réseau est alors sur 24 bits et les identifiants de machine sont sur 8 bits (au plus 254 machines par sous-réseau, numérotées de 1 à 254) ;

Masque de sous-réseau

Un masque de sous-réseau est une donnée sur 4 octets qui, avec l'adresse du sous-réseau, caractérise les IP du sous-réseau.

Un bit du masque de sous-réseau est à 1 si pour toutes les adresses IP du sous-réseau, ce même bit est le même pour l'adresse IP et le sous-réseau.

Par exemple, pour le réseau de classe A 37.0.0.0 avec le masque de sous-réseau 255.0.0.0, les 8 premiers bits de toutes les adresses IP du sous-réseau valent 37.

Autre exemple : pour le sous-réseau de classe C 193.56.49.0 et le masque de sous-réseau 255.255.255.0, les 24 premiers bits de toutes les IP du sous-réseau sont 193.56.49.

On peut désigner un sous-réseau par son adresse et son masque, mais on peut aussi désigner le sous-réseau en donnant seulement le nombre de bits du masque. On parle alors, pour reprendre les deux exemples précédents, du sous-réseau 37.0.0.0/8 ou du sous-réseau 193.56.49.0/24.

Routage

Le routage permet de faire communiquer plusieurs sous-réseaux. Une passerelle (en anglais *gateway*) est en communication avec différents sous-réseaux sur différentes interfaces, et assure la communication entre les différents sous-réseaux (voir figure 8.1).

Routes

Une route définie sur une station est un chemin que doivent prendre les paquets à destination d'un certain sous-réseau. Prenons l'exemple (voir figure 8.1) d'une station, appelée station 1, d'adresse IP 112.65.77.8 sur un réseau 112.0.0.0/8.

(Fig. 8.1: Exemple de passerelle faisant communiquer deux réseaux)

Elle est connectée à une passerelle qui a pour IP dans ce réseau 112.65.123.3 sur son interface eth0. La passerelle est aussi connectée au réseau 192.168.0.0/24 via son interface eth1 qui a pour IP 192.168.0.1. Si la station 1 veut communiquer directement avec la station, appelée station 6, d'adresse IP 192.168.0.2 sur le réseau 192.168.0.0/24, trois condition doivent être réunies ;

- Une route doit être définie sur la station 1 indiquant que les paquets à destination du réseau 192.168.0.0/24 doivent passer par la passerelle 112.65.123.3. Pour cela, on peut utiliser la commande route :

```
route add -net 192.168.0.0/24 gw 112.65.123.3
```

- Une route doit être définie sur la station 6 indiquant que les paquets à destination du réseau 112.0.0.0/8 doivent passer par la passerelle 192.168.0.1 ; Pour cela, on peut utiliser la commande route :

```
route add -net 112.0.0.0/8 gw 192.168.0.1
```

- La passerelle doit être configurée pour transmettre (ou *forwarder*) les paquets IP d'un réseau à l'autre, ce qui se fait par l'une des commandes suivantes (les deux font la même chose, inutile donc de se répéter) :

```
echo 1 >/proc/sys/net/ipv4/ip_forward
sysctl -w net.ipv4.ip_forward=1
```

Remarque : Il faut refaire ces commandes après un redémarrage. Pour éviter de relancer ces commandes manuellement, on peut les mettre dans des scripts d'initialisation au démarrage avec la commande update-rc.d (sous debian)). Pour ajouter une script my_ script à l'initialisation :

```
mv my_script /etc/init.d
update-rc.d my_script defaults
```

Si l'IP-forwarding est activé via uns sysctl, le fichier standard de configuration de ce paramètre est `/etc/sysctl.conf`, où la ligne est déjà en commentaire.

On peut voir l'état des routes avec la commande route -n. Par exemple, sur la station 1 :

```
route -n
Destination           Gateway                 Genmask               Flags Metric Ref     Use Iface
192.168.0.0           112.65.123.3            255.255.255.0         U     0       0         0 eth2
etc...
```

Sur la station 6

```
route -n
Destination           Gateway                 Genmask               Flags Metric Ref     Use Iface
112.0.0.0             192.168.0.1             255.0.0.0             U     0       0         0 wlan0
etc...
```

Pour supprimer une route, par exemple vers le réseau 193.86.46.0/24 via une passerelle 196.24.52.1, on fait :

```
route del -net 193.86.46.0/24 gw 196.24.52.1
```

Route par défaut (gateway)

Quand on a défini un certain nombre de routes sur une station, on peut définir une route spéciale pour les paquets IP à destination des réseaux non prévus dans les autres routes. On appelle une telle route une route par défaut. En général, c'est la route qu'il faut employer pour aller sur internet. On emploie le réseau 0.0.0.0 (masque 255.255.255.255). Pour définir une route par défaut on peu employer route. Par exemple, pour définir la route par défaut via la passerelle 194.56.87.1 :

```
route add default gw 194.56.87.1
```

Pour supprimer cette même route :

```
route del default gw 194.56.87.1
```

NAT et masquerading

Lorsqu'un hôte ayant une adresse IP sur un réseau local cherche à se connecter sur un réseau plus vaste, par exemple sur internet, via une passerelle, cet hôte a besoin d'une adresse IP sur le réseau vaste. Pour cela, soit on demande à ce que les adresses du réseau local soient routées sur le réseau global, mais il faut alors demander à réserver un plage d'adresses sur le réseau global, soit l'administrateur de la passerelle a la possibilité de prêter l'IP de la passerelle aux machines du réseau local. Pour cela, on utilise iptables avec NAT. Par exemple, si la passerelle se connecte à internet via son interface eth0, il suffit d'exécuter la commande suivante sur la passerelle :

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Toute machine du réseau local (par exemple 192.168.0.0/24) qui se connecte à internet via cette passerelle aura alors l'adresse IP de la passerelle sur internet. On peut aussi donner aux machines du réseau local une autre adresse IP que l'on spécifie avec –to :

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 193.56.17.9
```

Nous aurons un aperçu plus complet des possibilités d'iptables dans la documentation sur Netfilter.

Redirection

Il est également possible de modifier la destination d'un paquet. C'est utile lorsqu'on veut cacher une machine derrière une autre : le client se connecte à un serveur, qui va transmettre tous les paquets à un autre. Pour le client c'est transparent, c'est comme si c'était le premier serveur qui répondait. On appelle cela du *Destination NAT* (DNAT).

Exemple pour rediriger toutes les connexions du port 22 vers le serveur 192.168.1.12 :

```
iptables -t nat -A PREROUTING -p tcp --dport 22 -j DNAT --to 192.168.1.12
```

Sources et contributeurs de l'article

Administration réseau sous Linux/Configuration réseau *Source*: http://fr.wikibooks.org/w/index.php?oldid=368856 *Contributeurs*: Geek hippie, Michael Witrant, 4 modifications anonymes

Administration réseau sous Linux/NFS *Source*: http://fr.wikibooks.org/w/index.php?oldid=347047 *Contributeurs*: Michael Witrant, Tavernier, 5 modifications anonymes

Administration réseau sous Linux/Samba *Source*: http://fr.wikibooks.org/w/index.php?oldid=359813 *Contributeurs*: Michael Witrant, 5 modifications anonymes

Administration réseau sous Linux/Apache *Source*: http://fr.wikibooks.org/w/index.php?oldid=419487 *Contributeurs*: JackPotte, Michael Witrant, TouzaxA, 5 modifications anonymes

Administration réseau sous Linux/ProFTPD *Source*: http://fr.wikibooks.org/w/index.php?oldid=413414 *Contributeurs*: Michael Witrant, Sasa, 4 modifications anonymes

Administration réseau sous Linux/DHCP *Source*: http://fr.wikibooks.org/w/index.php?oldid=360469 *Contributeurs*: DavidL, Michael Witrant, 5 modifications anonymes

Administration réseau sous Linux/Netfilter *Source*: http://fr.wikibooks.org/w/index.php?oldid=414141 *Contributeurs*: Michael Witrant, Sasa, TouzaxA, 5 modifications anonymes

Administration réseau sous Linux/TCP Wrapper *Source*: http://fr.wikibooks.org/w/index.php?oldid=414036 *Contributeurs*: JackPotte, Michael Witrant, 5 modifications anonymes

Administration réseau sous Linux/Tcpdump *Source*: http://fr.wikibooks.org/w/index.php?oldid=417844 *Contributeurs*: Michael Witrant, 3 modifications anonymes

Administration réseau sous Linux/SSH *Source*: http://fr.wikibooks.org/w/index.php?oldid=419551 *Contributeurs*: Michael Witrant, 6 modifications anonymes

Administration réseau sous Linux/Routage *Source*: http://fr.wikibooks.org/w/index.php?oldid=417769 *Contributeurs*: DavidL, JackPotte, Michael Witrant, Mro, Perditax, Rome2, Savant-fou, 10 modifications anonymes

Source des images, licences et contributeurs

Image:Netfilter schema - Filter table only.png *Source*: http://fr.wikibooks.org/w/index.php?title=Fichier:Netfilter_schema_-_Filter_table_only.png *Licence*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributeurs*: Michael Witrant

Licence