Administration Systèmes & Réseaux

Administration et intégration du réseau dans un système

May 3, 2015

Houcemeddine HERMASSI

houcemeddine.hermassi@enit.rnu.tn

École Nationale d'Ingénieurs de Carthage ENI-CAR
Université Carthage
Tunisie

Plan de cour

Rappel TCP/IP

le filtrage de paquets réseau

Services réseaux xinetd

Bases

L'origine de **TCP/IP** provient des recherches du **DARPA** (Defense Advanced Research Project Agency) qui débutent en 1970 et débouchent sur **ARPANET**. Dans les faits, le DARPA a financé l'université de Berkeley qui a intégré les protocoles de base de TCP/IP au sein de son système **UNIX BSD 4**.
TCP/IP s'est popularisé grâce à son interface générique de programmation d'échanges de données entre les machines d'un réseau, les primitives **sockets**, et l'intégration de protocoles applicatifs. Les protocoles de TCP/IP sont supervisés par l'**IAB** (Internet Activities Board) lui-même supervisant deux autres organismes :

- L'**IRTF** (Internet Reseach Task Force) qui est responsable du développement des protocoles.
- L'**IETF** (Internet Engineering Task Force) qui est responsable du réseau Internet.

Les adresses réseau sont distribuées par le **NIC** (Network Information Center) et en France l'**INRIA**. L'ensemble des protocoles de TCP/IP est décrit dans les documents **RFC** (Request For Comments) (voir le RFC 793).

- La couche inférieure est **IP** (Internet Protocol).
- La couche de transport est **TCP** (Transmission Control Protocol) ou **UDP**(User Datagram Protocol).
- Les couches supérieures sont les couches des protocoles applicatifs, par exemple:
  - **NFS** (Network File System) : partage de fichiers à distance.
  - **DNS** (Domain Name System) : association hôte<->IP.
  - **FTP** (File Transfer Protocol) : transfert de fichiers.
  - **TELNET** : émulation d'un terminal de type texte...

Classes

Il est important de savoir avant l'installation dans quel type de réseau doit s'intégrer le nouveau serveur, TCP/IP bien sûr, mais il faut déjà lui réserver une adresse IP, un hostname (nom de machine réseau), connaître les diverses passerelles, le nom de domaine, la classe utilisée et le masque de sous-réseau netmask.
Voici un bref rappel sur les classes IP. Une adresse IP est définie sur 32 bits et représentée par quatre nombres séparés par des points : **n1.n2.n3.n4**. Cette adresse est constituée de deux parties qui définissent l'adresse réseau et l'hôte dans le réseau.
Distinguez, suivant les cas, quatre ou cinq classes d'adresses : A, B, C, D et E, mais seules les trois premières nous intéressent.
Légende : N et h sont des bits, N identifiant du réseau h identifiant de la machine.

```
Classe A : 0NNNNNNN hhhhhhhh hhhhhhhh hhhhhhhh soit 1.x.x.x à "
126.x.x.x."
n1 est compris entre 1 et 126."
16777214 hôtes, 127 réseaux."
Classe B : 10NNNNNN NNNNNNNN hhhhhhhh hhhhhhhh soit de 128.0.x.x à "
191.255.x.x."
n1 est compris entre 128 et 191. "
65534 hôtes, 16382 réseaux."
Classe C : 110NNNNN NNNNNNNN NNNNNNNN hhhhhhhh soit de 192.0.0.x à "
223.255.255.x."
n1 est compris entre 192 et 223. "
254 hôtes, 2097150 réseaux."
Classe D : Commence par 1110, pour la multidiffusion IP. "
Classe E : Commence par 1111, pour expérimentation."
```

Classes

Il existe des adresses d'hôtes qui ne peuvent pas être exploitées. Par exemple dans la classe C on ne peut avoir que 254 hôtes, alors que l'identifiant de la machine est codé sur 8 bits (donc 256 valeurs). C'est que l'adresse 0 représente l'adresse du réseau, et l'adresse 255 celle du **broadcast** (multidiffusion).
Notez que les adresses suivantes ne doivent pas être routées sur Internet et sont réservées aux réseaux locaux.

- 10.0.0.0 - 10.255.255.255 (10/8)
- 172.16.0.0 - 172.31.255.255 (172.16/12)
- 192.168.0.0 - 192.168.255.255 (192.168/16)

L'adresse 127.0.0.1 est l'adresse de loopback ou bouclage : elle représente la machine elle-même, ainsi que le sous-réseau127.0.0.0/8.

sous-réseaux

De plus, il est possible de découper ces réseaux en sous-réseaux à l'aide de masques permettant un découpage plus fin des adresses. Un **netmask** est un masque binaire qui permet de séparer immédiatement l'adresse du réseau et du sous-réseau de l'adresse de l'hôte dans l'adresse IP globale. Les masques prédéfinis sont :

- **Classe A** : 255.0.0.0
- **Classe B** : 255.255.0.0
- **Classe C** : 255.255.255.0

Pour communiquer directement entre eux les hôtes doivent appartenir à un même réseau ou sous-réseau. Calculer un sous-réseau est assez simple. Voici un exemple pour un réseau de classe C.

- Réseau : 192.168.1.0
- Adresse de réseau : 192.168.1.255
- Masque de réseau : 255.255.255.0

sous-réseaux

Calculer un masque de sous-réseau :

- Pour calculer le masque de sous-réseau, vous devez tout d'abord déterminer combien de machines vous souhaitez intégrer dans celui-ci. Un réseau de classe C permet d'intégrer 254 machines (0 et 255 étant réservés). Vous souhaitez créer des réseaux contenant 60 machines. Ajoutez 2 à cette valeur pour les adresses réservées (adresse du sous-réseau et adresse de broadcast) ce qui donne **62** .
- Une fois le nombre de machines déterminé, trouvez la puissance de deux exacte ou juste supérieure au nombre trouvé. 2 puissance 6 donne **64** .
- Écrivez le masque en binaire, placez tous les bits du masque de réseau de classe C à 1 et placez à 0 les 6 premiers bits du masque correspondant à la partie machine : **1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 11000000**
- Convertissez ce masque en décimal : 255.255.255.192, et calculez l'ensemble des sous-réseaux possibles. Comme vous êtes dans un réseau de classe C, vous pouvez encore faire varier les deux derniers bits de la partie machine :
  - 00xxxxxx : 255.255.255.0
  - 01xxxxxx : 255.255.255.64
  - 10xxxxxx : 255.255.255.128
  - 11xxxxxx : 255.255.255.192
- Au final, vous obtenez quatre sous-réseaux de 62 machines, soit 248 machines. Vous tombez bien sur 256 si vous rajoutez les quatre adresses de broadcast et les quatre adresses de réseau.

Routage

Le masque de réseau permet de déterminer si une machine destinataire est sur le même réseau que vous ou non. Il faut indiquer le chemin que doivent prendre les paquets IP pour rejoindre leur destination. Si votre machine est un poste client disposant d'une seule carte réseau et que ce réseau ne comporte qu'un seul routeur (cas classique d'une connexion vers Internet) alors vous devez créer deux routes. La première est celle indiquant quelle carte réseau doivent emprunter les paquets pour accéder au reste du réseau (au sous-réseau), la seconde quelle route doivent emprunter les paquets pour sortir du réseau. Généralement, on parle de route par défaut quand un seul routeur est présent.

- Vers réseau1 -> utiliser interface réseau gauche.
- Vers réseau2 -> utiliser interface réseau droite.
- Vers autres -> utiliser interface réseau droite vers routeur1.

Exemple : Réseau1 de classe C 192.168.1.0 sur eth0, Réseau2 de classe B 172.16.0.0 sur eth1, adresse de routeur 192.168.1.254.
Tous les paquets réseaux vers 192.168.1.0 transiteront par eth0. Tous les paquets à destination de 172.16.0.0 transiteront par eth1. Par défaut, tous les autres paquets pour les réseaux non spécifiés transiteront par eth0 et seront traités par la passerelle 192.168.1.254 qui routera les paquets.

Cas des distributions de type Red Hat/Fedora

Red Hat propose des outils pour configurer le réseau de base sans passer par la manipulation des fichiers de configuration. Le programme netconfig est en mode texte et permet de mettre en place la configuration TCP/IP de base (IP statique ou dynamique, routeur, nom d'hôte, serveur de noms).

```
# netconfig -device eth0"
```

La commande graphique **system-config-network** lance une interface plus complète.

Red Hat/Fedora: Interfaces réseau

La configuration de base d'une interface réseau se fait à l'aide de la commande **ifconfig**. Cependant la plupart des distributions utilisent des scripts d'administration et des fichiers de configuration qui simplifient énormément les choses car ils configurent à la fois l'interface réseau et les routes.

1- Configuration de eth0 pour l'adresse de classe C 192.168.1.2

```
ifconfig eth0 inet 192.168.1.2 netmask 255.255.255.0"
ifconfig eth0 192.168.1.2"
```

2- Activation de l'interface réseau eth0 :

```
ifconfig eth0 up "
```

3- Arrêt de l'interface réseau eth0 :

```
ifconfig eth0 down "
```

4- Affichage des informations de eth0 :

```
# ifconfig eth0 "
```

5- Affichage de toutes les interfaces réseaux activées :

```
ifconfig "
```

6- Affichage de toutes les interfaces réseaux activées ou non :

```
ifconfig -a "
```

Le mieux reste d'utiliser les scripts **ifup** et **ifdown**. Ceux-ci se basent sur les fichiers présents dans /etc/sysconfig/network-scripts/. Ces fichiers de configuration d'interface se nomment ifcfg-xxx où xxx est le nom de l'interface réseau, comme eth0 :

```
DEVICE=eth0"
IPADDR=192.168.1.2"
NETMASK=255.255.255.0"
NETWORK=192.168.1.0"
BROADCAST=192.168.1.255"
ONBOOT=yes"
```

Red Hat/Fedora: Paramètres généraux

Le fichier `/etc/sysconfig/network"` contient les paramètres généraux du réseau.

```
NETWORKING=yes "
HOSTNAME=poste1.monreseau.org # nom complet"
GATEWAY=0.0.0.0 # passerelle par défaut"
NISDOMAIN= # nom du domaine NIS"
```

- **NETWORKING** : activation ou non du réseau.
- **HOSTNAME** : nom de domaine complet FQDN.
- **GATEWAY** : adresse IP de la passerelle.
- **GATEWAYDEV** : interface réseau permettant d'accéder à la passerelle.
- **NISDOMAIN** : cas d'un domaine NIS..

Machines de type Debian

Le fichier de configuration des interfaces réseaux sous Debian (et Ubuntu) est situé dans **/etc/network/interfaces**. Il n'a pas du tout le même format que précédemment:

```
# cat interfaces"
```

Cet exemple montre trois types d'interfaces:

- l'interface lo de loopback
- l'interface eth1 en dhcp, ne nécessitant pas de configuration plus avancée
- l'interface eth0 configurée statiquement.

La syntaxe générale d'une déclaration est la suivante:
`interface nom type mode"` Avec une configuration statique, précisez les différents paramètres avec les mots clés suivants:

- address: l'adresse IP.
- netmask: le masque de sous-réseau.
- broadcast: l'adresse de broadcast.
- gateway: la passerelle par défaut.

Le fichier /**etc/hostname** contient le nom de la machine:

```
#cat /etc/hostname"
```

Machines Debian: Routage

Avec l'utilisation des fichiers et des commandes précédentes, il n'y a pas besoin de créer un routage spécifique car la passerelle par défaut est déjà présente (cf. paramètres généraux) et les routes pour les interfaces réseaux sont automatiquement mises en place par **ifup**. Cependant on peut utiliser la commande **route**.

1- Affiche les routes actuelles :

```
route"
netstat -nr"
```

2- Ajout de l'entrée loopback:

```
route add -net 127.0.0.0"
```

3- Ajoute la route vers le réseau 192.168.1.0 passant par eth0. Netmask peut être omis.

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0"
```

4- Ajoute la passerelle par défaut vers le routeur :

```
route add default gw 192.168.1.254 "
```

5- Supprime la route vers le réseau 172.16.0.0 :

```
route del -net 172.16.0.0 eth0"
```

Ping

La commande **ping** est une commande centrale, voire incontournable. La première chose que l'on fait généralement pour savoir si une machine est accessible ou non, c'est d'essayer de la « pinguer » (sous réserve que la configuration du firewall autorise les requêtes ICMP).
Ping émet un « écho » réseau, un peu comme un sonar, et attend une réponse, le retour de l'écho. Il utilise pour cela le protocole ICMP. Interrompez la commande ping avec [Ctrl] C.

```
$ ping www.kde.org "
```

Trois paramètres doivent attirer votre attention :

- -c permet de préciser le nombre d'échos à émettre.
- -b permet d'émettre un écho sur une adresse de broadcast
- -I permet de spécifier l'interface réseau.

Dans le premier cas le paramètre peut être utile pour tester dans un script si un serveur répond :

```
# ping -c 1 10.9.238.170 >/dev/null 2>&1 && echo Ïe serveur répond¨
```

Dans le second cas, toutes les adresses du sous-réseau concerné par l'adresse de broadcast doivent répondre.

```
# ping -b 192.168.1.255"
```

Dans le dernier cas, vous pouvez spécifier une carte de sortie. Cette option est très utile pour vérifier une résolution DNS ou une route.

```
# ping -I eth0 192.168.1.60"
```

Traceroute

Quand vous tentez d'accéder à un hôte distant depuis votre machine, les paquets IP passent souvent par de nombreuses routes, parfois différentes selon le point de départ et de destination, l'engorgement, etc. Le trajet passe par de nombreuses passerelles (gateways), qui dépendent des routes par défaut ou prédéfinies de chacune d'elles.
La commande **traceroute** permet de visualiser chacun des points de passage de vos paquets IP à destination d'un hôte donné. Dans l'exemple suivant, l'hôte situé en région parisienne sur le réseau du fournisseur Free tente de déterminer la route empruntée pour se rendre sur le serveur www.kde.org. L'adresse IP source (hors réseau local) est volontairement masquée.

```
$ traceroute www.kde.org "
```

Whois

Savez-vous que vous pouvez obtenir toutes les informations voulues sur un domaine (toto.fr) à l'aide de la commande **whois** ? Par exemple, pour obtenir toutes les informations sur le domaine kde.org :

```
whois kde.org"
```

Netstat

La commande netstat permet d'obtenir une foule d'informations sur le réseau et les protocoles.
Le paramètre -i permet d'obtenir l'état des cartes réseaux, afin de déterminer une éventuelle panne ou un problème de câble :

```
# netstat -i "
```

Si vous rajoutez le paramètre -e, vous obtenez le même résultat qu'avec ifconfig -a.

```
# netstat -ei"
```

IPTraf

La commande **iptraf** permet de visualiser en temps réel l'activité du réseau via un outil texte interactif ou non (ligne de commande). Les menus sont clairs. Vous vous y déplacez avec les touches fléchées et les divers raccourcis précisés.
La commande suivante montre l'affichage détaillé des statistiques de trafic de la carte eth0. Cet écran est accessible via la ligne de commande avec :

```
# iptraf -d eth0"
```

le filtrage de paquets réseau

Netfilter

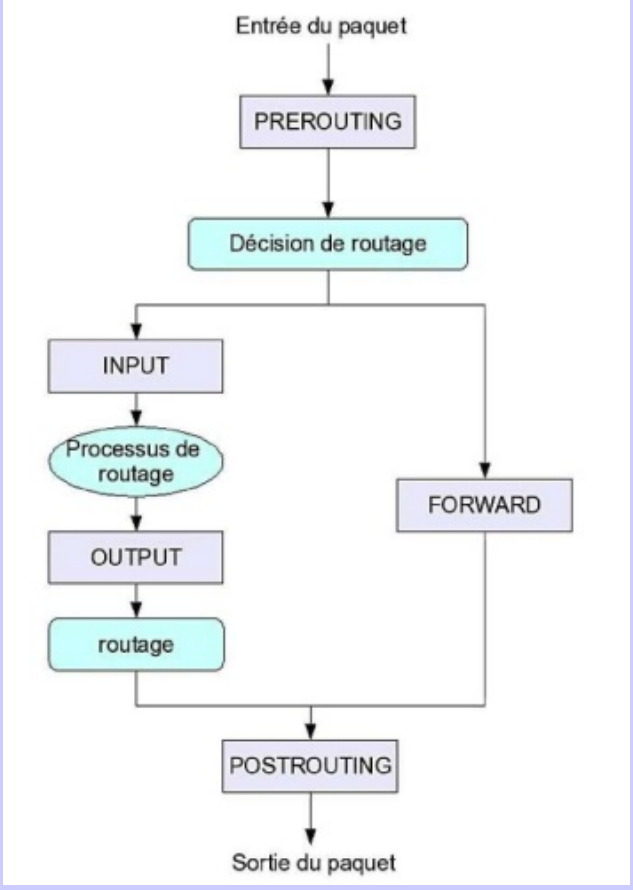
Présentation

Netfilter est une architecture de filtrage des paquets pour les noyaux Linux 2.4 et 2.6. Le filtrage se fait au sein même du noyau au niveau des couches 2, 3 et 4 du modèle OSI, c'est-à-dire les liaisons données, réseau et transport. Il est par exemple capable d'agir à bas niveau sur les interfaces ethernet (2), sur la pile IP (3) et sur les protocoles de transport comme TCP (4). Le filtrage est **stateless** : comme Netfilter n'inspecte que les en-têtes des paquets, il est extrêmement rapide et n'entraîne pas de temps de latence.
L'inspection des contenus des paquets (protocoles applicatifs) peut se faire à l'aide d'extensions mais devrait cependant être réservée à des outils de l'espace utilisateur.
Autrement dit, Netfilter est un firewall agissant au niveau du protocole.
Le programme utilisateur permettant d'agir sur les règles de filtrage est **iptables**.
L'implémentation au niveau du noyau est réalisée par des modules.

le filtrage de paquets réseau
Netfilter

Vie d'un paquet

Plutôt qu'un long discours, voici un schéma. Le paquet arrive par le haut et ressort par le bas. Entre les deux, il passe par différents états au niveau de netfilter.



le filtrage de paquets réseau
Netfilter

Vie d'un paquet

Chaque état (rectangle) correspond à un point de filtrage possible par la commande **iptables**.

- **PREROUTING** : traite les paquets à leur arrivée. Si un paquet est à destination du système local, il sera traité par un processus local (INPUT, OUTPUT). Sinon, et si le forwarding est activé, les règles FORWARD et POST_ROUTING seront appliquées.
- **FORWARD** : les paquets ne font que traverser le système local. Traite les paquets routés à travers le système local.
- **INPUT** : traite les paquets destinés au système local, en entrée (après le routage).
- **OUTPUT** : traite les paquets quittant le système local, avant POSTROUTING.
- **POSTROUTING** : traite les paquets juste avant leur sortie du système.

le filtrage de paquets réseau
Netfilter

Principe des règles

Lorsqu'un paquet est traité par netfilter, il l'est par rapport à un certain nombre de règles qui déterminent ce qu'il faut en faire.

- Les règles sont ordonnées : la position d'une règle dans une liste indique quand et si la règle sera utilisée.
- Les paquets sont testés avec chacune des règles, l'une après l'autre.
- Netfilter fonctionne selon le mécanisme de la première correspondance. Si une règle correspond, les autres règles sont négligées et la vérification s'arrête.
- Une règle peut spécifier plusieurs critères.
- Pour qu'une règle corresponde à un paquet, tous les critères doivent correspondre.
- Si malgré toutes les règles, le paquet passe, une règle par défaut peut être appliquée.

le filtrage de paquets réseau
Netfilter

Cible de règles

Une cible de règle détermine quelle est l'action à entreprendre lorsqu'un paquet correspond aux critères d'une règle. On utilise l'option -j de **iptables** pour spécifier la cible.
Les deux cibles de base sont DROP et ACCEPT. Des extensions de netfilter ajoutent d'autres cibles comme LOG ou REJECT.

- **DROP** : le paquet est rejeté. Aucune notification n'est envoyée à la source.
- **REJECT** : le paquet est rejeté, retournant une erreur à la source.
- **ACCEPT** : le paquet est accepté.
- **LOG** : une information est envoyée à syslog pour les traces.

Vous pouvez créer des règles sans cible. Dans ce cas, la règle incrémentera un compteur de paquets et un compteur d'octets associés à la règle afin d'établir une collecte de statistiques.

le filtrage de paquets réseau
Netfilter

Premier exemple

Voici une règle simple qui interdit tous les paquets en provenance de 192.168.1.11.
Les deux cibles de base sont DROP et ACCEPT. Des extensions de netfilter ajoutent d'autres cibles comme LOG ou REJECT.

```
# iptables -A INPUT -s 192.168.1.11 -j DROP"
```

- **-A** : point de filtrage (INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING) qu'on appelle aussi **chaîne**.
- **-s** : source, peut être une adresse IP, un nom d'hôte, un réseau, etc.
- **-j** : jump, cible de la règle (ACCEPT, DROP, REJECT...)

Résultat : vous interdisez l'entrée des paquets dont la source est 192.168.1.11.

le filtrage de paquets réseau
Netfilter

Opérations de base

Les règles sont numérotées à partir de 1.
1- Ajoutez une règle avec **-A**.

```
# iptables -A INPUT -s 192.168.1.2 -j DROP "
```

2- Insérez une règle avec **-I** à la position souhaitée. .

```
# iptables -I OUTPOUT -d 192.168.1.25 -j DROP 3 # inserer à la 3eme position"
```

3- Supprimez une règle avec **-D**.

```
# iptables -D INPUT 1  supprime la règle 1 "
```

4- Listez les règles avec **-L**.

```
# iptables -L OUTPUT "
# iptables -L"
```

5- Utilisez **-F** (flush) pour supprimer l'ensemble des règles.

```
# iptables -F "
```

6- Utilisez **-P** (policy) pour modifier les règles par défaut d'une chaîne.

```
# iptables -P INPUT DROP "
# iptables -L INPUT "
```

le filtrage de paquets réseau
Netfilter

Critère de correspondance: Général

Les critères de correspondance déterminent la validité d'une règle. Tous les critères doivent être vérifiés pour qu'un paquet soit bloqué. Les critères de base sont :

- **-i** : interface entrante (filtrage couche 2)
- **-o** : interface sortante (filtrage couche 2)
- **-p** : protocole de couche 4. Pour les noms, voir le fichier `/etc/protocols`
- **-s** : adresse IP de la source (ou réseau)
- **-d** : adresse IP de destination (ou réseau).

1- Interdire les entrées par eth0.

```
-A INPUT -i eth0 -j DROP"
```

2- Interdire le forward entre eth1 et eth2.

```
-A FORWARD -i eth1 -o eth0 -j DROP "
```

3- Interdire le protocole ICMP en entrée (le ping !) .

```
iptables -A input -p icmp -j DROP "
```

le filtrage de paquets réseau
Netfilter

Critère de correspondance: TCP, UDP et ICMP

Suivant le protocole (couche 4) certaines options sont possibles. C'est le cas de tcp, udp ou icmp (notamment utilisé par ping). Le filtrage au niveau des protocoles est généralement effectué par des extensions à netfilter.

- **-p** : protocole (tcp, udp, icmp, etc.)
- **–sport** : port source
- **–dport** : port destination

Si vous souhaitez par exemple interdire les connexion entrantes à destination du port 80 (serveur httpd) procédez ainsi :

```
iptables -A INPUT -p tcp -dport 80 -j DROP "
```

Critère de correspondance: Arguments des critères

Pour les adresses, vous pouvez spécifier :

- un hôte par son nom ou son adresse IP
- un réseau par son nom ou son masque (192.168.1.0/24, 192.168.1.0/255.255.255.0). Pour les ports :
  - un numéro
  - un nom (voir /etc/services)
  - une gamme de ports : **123:1024**.

Dans tous les cas, la négation avec ! (point d'exclamation) est possible.
Pour interdire en entrée toutes les connexions sauf celles de 10.0.0.1 :

```
# iptables -A INPUT -s ! 10.0.0.1 -j DROP "
```

Critère de correspondance: Sauver ses réglages

Les règles définies avec iptables sont chargées en mémoire et transmises dynamiquement au noyau. En redémarrant la machine, elles sont perdues. Red Hat permet de sauver l'ensemble des règles de manière à les rendre persistantes.

```
# service iptables save "
```

Les règles sont sauvées dans le fichier **/etc/sysconfig/iptables**.

Présentation

Le démon **xinetd** est un « super-service » permettant de contrôler l'accès à un ensemble de services, telnet par exemple. Beaucoup de services réseaux peuvent être configurés pour fonctionner avec xinetd, comme les services ftp, ssh, samba, rcp, http, etc. Des options de configuration spécifiques peuvent être appliquées pour chaque service géré.
Lorsqu'un hôte client se connecte à un service réseau contrôlé par xinetd, xinetd reçoit la requête et vérifie tout d'abord les autorisations d'accès TCP (voir **tcp_wrappers** au prochain chapitre) puis les règles définies pour ce service (autorisations spécifiques, ressources allouées, etc.). Une instance du service est alors démarrée et lui cède la connexion. À partir de ce moment **xinetd** n'interfère plus dans la connexion entre le client et le serveur.

Configuration

Les fichiers de configuration sont :

- **/etc/xinetd.conf** : configuration globale
- **/etc/xinetd.d/*** : répertoire contenant les fichiers spécifiques aux services. Il existe un fichier par service, du même nom que celui précisé dans /etc/services.

```
$ ls -l /etc/xinetd.d "
```

1- Contenu de xinetd.conf :

```
defaults
{
instances = 60
log_type = SYSLOG authpriv
log_on_success = HOST PID
log_on_failure = HOST
cps = 25 30
}
includedir /etc/xinetd.d
```

- **instances** : nombre maximal de requêtes qu'un service xinetd peut gérer à un instant donné.
- **log_type** : dans notre cas, les traces sont gérées par le démon **syslog** via **authpriv** et les traces sont placées dans /var/log/secure. **FILE /var/log/xinetd** aurait placé les traces dans /var/log/xinetd.
- **log_on_success** : xinetd va journaliser l'événement si la connexion au service réussit. Les informations tracées sont l'hôte (**HOST**) et le **PID** du processus serveur traitant la connexion.
- **log_on_failure** : idem mais pour les échecs. Il devient simple de savoir quels hôtes ont tenté

Configuration

Exemple /etc/xinetd.d/telnet :

```
# default: on "
# description: The telnet server serves telnet sessions; it uses  "
# unencrypted username/password pairs for authentication."
service telnet
{
disable = no
flags = RUSE
socket type = stream
wait = no
user = root
server =/usr/sbin/in.telnetd
log on failure +=USERID
}
```

La première ligne en commentaire, **default**, a une importance particulière. Elle n'est pas interprétée par xinetd mais par **ntsysv** ou **chkconfig** pour déterminer si le service est actif.

- **service** : nom du service qui correspond à un service défini dans /etc/services.
- **flags** : attributs pour la connexion. REUSE indique que la socket sera réutilisée pour une connexion telnet.
- socket_type : spécifie le type de socket. Généralement stream (tcp) ou dgram (udp). Une connexion directe IP se fait par raw.
- **wait** : indique si le serveur est single-threaded (yes) ou multi-threaded (no).
- **user** : sous quel compte utilisateur le service sera lancé.

Démarrage et arrêt des services

On distingue deux cas.
Premier cas, le service **xinetd** est un service comme un autre dont le démarrage ou l'arrêt peut s'effectuer avec la commande **service** ou directement via l'exécution de /etc/init.d/xinetd.

```
# service xinetd start "
```

Dans ce cas, la commande **chkconfig** (Red Hat, openSUSE) autorise ou non le lancement du service au démarrage pour chaque niveau d'exécution (runlevel).

```
# chkconfig -level 345 xinetd on "
```

Second cas, comme xinetd gère plusieurs services, l'arrêt de xinetd arrête tous les services associés, et le démarrage de xinetd lance tous les services associés. Il n'est pas possible de choisir quels services de xinetd seront lancés dans tel ou tel niveau d'exécution. Mais vous pouvez choisir d'activer ou de désactiver simplement un service avec chkconfig.

```
# chkconfig telnet on "
```

Merci pour votre attention!