Licence 3 Informatique - SR2

Systèmes et Réseaux II

Chapitre 4

Services Réseaux
Partie I : lancement et services généraux

Éric Leclercq
Département IEM / uB
Eric.Leclercq@u-bourgogne.fr
Bureau R8 Aile H

26 janvier 2024

Plan du chapitre

Notion de services et démons

Définition :

Les services sont des processus qui permettent à un utilisateur distant de devenir utilisateur (réel ou virtuel) d'une machine UNIX et par conséquent de l'utiliser (éventuellement avec des ressources restreintes)

- Les services sont à l'écoute des connexions sur ports TCP/IP
- Chaque port réseau (codé sur 16 bits) est associé à un service particulier, c'est ce qui permet au noyau de délivrer les paquets aux processus concerné
- L'utilisateur distant établit une connexion réseau avec la machine serveur puis le processus serveur (service ou démon), à l'écoute du port (enregistré dans le noyau), accepte la connexion et traite la requête (éventuellement en se dupliquant)

Notion de démon

- Le mot démon vient du mot anglais daemon qui lui-même est un acronyme signifiant Disk And Execution MONitor

Définition :

un processus qui s'exécute en arrière-plan plutôt que sous le contrôle direct d'un utilisateur.

- Les processus démons sont le plus souvent lancés lors de la phase de chargement du système d'exploitation
- Ils servent en général à répondre à des requêtes du réseau, à gérer l'activité du matériel ou à proposer des services à d'autres programmes
- De manière plus spécifique, sous un système UNIX, un démon peut être n'importe quel processus qui a le processus numéro 1 comme parent (`init`)

Notion de démon

- Tout processus dont le parent meurt sans attendre le statut de son processus enfant est adopté par `init`
- Pour lancer un démon fork off and die :
  - ”forker” une ou deux fois
  - faire arrêter le parent quand l'enfant à commencé ses opérations normales.

  Il existe également des processus qui ne peuvent pas être tués considérés comme les véritables démon du SE : X, inetd etc.

Deux modes de lancement des démons

- **autonome :** le démon écoute le port réseau désigné et lorsqu'il y a une connexion, il prend lui-même la connexion en charge pour fournir le service (exemple httpd)
- **esclave du démon inetd ou plus récement xinetd :** (x)inetd est spécialisé pour les connexions réseau
  - il possède un fichier de configuration qui indique quel programme doit être utilisé lorsqu'une connexion entrante est reçue
  - chacun des ports (service) doit être spécifié soit avec le protocole tcp, soit avec le protocole udp

  Quelque soit le mode de lancement pes ports standards sont décrits dans le fichier `/etc/services`.

Fichiers de configuration

Il y a deux fichiers essentiels sur lesquels vous êtes succeptibles d'intervenir :

- `/etc/services` qui enregistre les couples noms de service, numéros de port (**pour ajouter un protocole**)
- `/etc/xinetd.conf` ou `/etc/inetd.conf` qui sert pour la configuration du démon général et de ses services que nous développons dans la partie suivante (**pour lancer un nouveau service**

La sécurité de la machine dépend en premier lieu de ces deux fichiers

/etc/services

Le fichier `/etc/services` est base de données qui associe des noms compréhensibles par l'homme à des numéros de ports compréhensibles par la machine.

Son format est simple : fichier texte dont chaque ligne comprend trois champs séparés par des espaces ou tabulations

1. nom : mot qui identifie le service décrit
2. port/protocole : ce champ est divisé en deux
   - port : un nombre qui spécifie le numéro de port où le service désigné sera disponible. La plupart des services ont des numéros assignés (RFC-1340)
   - protocole : c'est soit tcp soit udp.
3. alias : autre nom utilisé pour désigner ce service

Extrait de /etc/services

Tout texte apparaissant après le caractère # est ignoré et traité comme commentaire.

```
echo            7/tcp
echo            7/udp
discard         9/tcp       sink null
discard         9/udp       sink null
systat          11/tcp      users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp      quote
chargen         19/tcp      ttytst source
chargen         19/udp      ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp      # SSH Remote Login Protocol
telnet          23/tcp
smtp            25/tcp      mail
```

Où en sommes nous ?

1. Lancement des services

2. **Les supers démons inetd/xinetd**
   - inetd
   - xinetd
   - xinetd : configuration

Le démon inetd

1. une requête arrive sur un port pris en charge par inetd
2. un processus est crée
3. la requête est transmise au processus

Il est possible d'installer un intermédiaire entre inetd et les processus : wrapper

Dans ce la requête est transmise au wrapper tcpd :

- tcpd décide de l'action à entreprendre relativement aux instructions contenues dans les fichiers hosts.{allow, deny}
- si la requête est autorisée, le démon associé la traite

Le démon inetd

Extrait du fichier de `/etc/inetd.conf`

```
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>

telnet  stream  tcp     nowait  telnetd.telnetd /usr/sbin/tcpd   /usr/sbin/in.telnetd
talk    dgram   udp     wait    nobody.tty      /usr/sbin/in.talkd  in.talkd
ntalk   dgram   udp     wait    nobody.tty      /usr/sbin/in.ntalkd in.ntalkd

smtp    stream  tcp     nowait   mail     /usr/sbin/exim exim -bs
imap2   stream  tcp     nowait   root     /usr/sbin/tcpd   /usr/sbin/imapd
imap3   stream  tcp     nowait   root     /usr/sbin/tcpd   /usr/sbin/imapd
```

Principe d'xinetd - extented Internet services daemon

www.xinetd.org Ses avantages :

- Fournit une excellente sécurité contre les intrusions
- Limite certains risques d'attaques par Deny of Services (DoS)
- Basé sur les techniques éprouvées du couple inetd + tcpd (tcp wrapper)
- Permet de fixer des règles d'accès à une machine, mais ses capacités s'étendent bien au-delà

Extensions fournies par xinetd

- contrôle d'accès pour des services TCP, UDP et RPC
- contrôle d'accès sur des plages horaires
- journalisation et limitations sur la taille des fichiers de log
- prévention contre les attaques de type Deny of Services (DoS)
- limitations du nombre de serveurs d'un même type
- limitations du nombre total de serveur
- attachement d'un service à une interface particulière
- peut faire office de proxy vers d'autres systèmes

xinetd : installation

La compilation à partir des fichiers sources et l'installation sont tout à fait classiques : `./configure ; make ; make install`
Néanmoins deux options particulières sont à considérer :

1. `--with-libwrap` : xinetd commencera par examiner les fichiers de configuration de tcpd (/etc/hosts.allow, deny) et ensuite, si l'accès est accordé, il utilisera ses propres mécanismes de contrôle
2. `--with-loadavg` : permet à xinetd de supporter l'option de configuration `max_load` (charge maximale). Cela permet de désactiver certains services lorsque la charge de la machine dépasse le seuil donné, ce qui est essentiel pour prévenir certains DoS

xinetd et inetd peuvent cohabiter, mais cela peut entraîner un comportement relativement imprévisible des deux démons

xinetd et les signaux :

xinetd a été programmé de manière à répondre aux signaux suivants :

- SIGUSR1 : reconfiguration soft, le fichier de configuration est relu et les paramètres des services sont ajustés ;
- SIGUSR2 : reconfiguration hard, comme ci-dessus, mais tue en plus les démons qui ne correspondent plus aux critères ;
- SIGTERM : termine xinetd et tous les démons qu'il a générés.
- SIGHUP : redemarre xinetd

structure du fichier /etc/xinetd.conf

Le fichier de configuration comprend :

- une section défaut dont les attributs seront utilisés pour tous les services pris en charge

```
defaults {
        attribut opérateur valeur(s)
        ...
}
```

- les sections spécifiques : autant de sections que de services désirés, chacune pouvant redéfinir des options qui lui seront spécifiques par rapport à celles par défaut

```
service nom_du_service {
        attribut opérateur valeur(s)
        ...
}
```

Exemple :

l'attribut `only_from` permet de donner une liste d'adresses autorisées à se connecter au serveur

```
only_from = 192.168.1.0/24 192.168.10.17
```

Si cet attribut figurent dans `defaults`, tous les services déclarés autorisent l'accès aux machines dont les adresses correspondent

Il est néanmoins possible de modifier ces valeurs par défaut au sein de chaque service

Les opérateurs

Trois opérateurs sont disponibles =, += et -= :

- = fixe une valeur à un attribut (la plupart des attributs ne supporte que l'opérateur =)
- += ajoute un élément à une liste de valeurs
- -= retire un élément

Les attributs (1)

- `flags` :
  - IDONLY : n'accepte de connexions que des clients qui disposent d'un serveur d'identification
  - NORETRY : évite que le processus soit à nouveau crée en cas d'échec
- `log_type` : xinetd utilise syslogd par défaut
  - SYSLOG sélecteur `[level]` : permet de choisir parmi les entrées daemon, auth, user ou local0-7 de syslogd ;
  - `FILE [taille_max [ taille_max_absolue ]]` : le fichier spécifié reçoit les informations. Les deux options fixent des limites de taille du fichier. La première provoque l'émission d'un message vers syslogd, la seconde arrête l'enregistrement des logs pour le service concerné

Les attributs (2)

- `log_on_success` : enregistrement des infos au démarrage du serveur
  - PID : le PID du serveur (service interne à xinetd, le PID vaut alors 0)
  - HOST : l'adresse du client
  - USERID : l'identité de l'utilisateur distant, (RFC1413)
  - EXIT : le statut de sortie du processus
  - DURATION : la durée de la session
- `log_on_failure`
  - HOST, USERID
  - ATTEMPT : enregistre le fait qu'une tentative d'accès a eu lieu
  - RECORD : enregistre toutes les informations disponibles sur le client

Les attributs (3)

- `no_access` : liste des clients dont on refuse les connexions à ce service
- `only_from` : liste des clients autorisés, si cet attribut n'a pas de valeur, l'accès à ce service est interdit
- `port` : port associé au service, si celui-ci est également défini dans `/etc/services` les ports doivent correspondre
- `protocol` : le protocole stipulé doit exister dans `/etc/protocols`, s'il n'en est pas fournit, le protocole par défaut associé à ce service est employé
- `server` : chemin vers le serveur à utiliser
- `cps` : limite le nombre de connexions entrantes, premier argument = nombre, lorsque ce seuil est dépassé, le service se désactive pour un délai, exprimé en secondes, fourni par le second argument

Les attributs (4)

- `instances` : nombre maximal de serveurs d'un même type pouvant fonctionner en même temps
- `max_load` : charge maximale pour un serveur, au-delà les requêtes sur ce serveur sont rejetées
- `per_source` soit un entier, soit UNLIMITED : restreint le nombre de connexions de la même origine
- `wait` comportement du service vis-à-vis des threads.
  - `yes` : le service est mono-thread, une seule connexion de ce type peut-être gérée à la fois par le service
  - `no` : à chaque nouvelle requête vers le service, un nouveau serveur est démarré par xinetd, dans la limite maximale définie

Exemple

```
jaromil root # cd /etc/xinetd.d/
jaromil xinetd.d # ls
cups-lpd  cvspserver  swat
jaromil xinetd.d # more *
 ::::::::::::::
cups-lpd
 ::::::::::::::
# default : off
# description : The cups-lpd mini daemon accepts jobs from a remote LPD client.
# $Header : /var/cvsroot/gentoo-x86/net-print/cups/files/cups.xinetd,v 1.3 2004/0
7/18 04 :18 :17 dragonheart Exp $

service printer
{
        socket_type = stream
        protocol    = tcp
        wait        = no
        user        = lp
        server      = /usr/lib/cups/daemon/cups-lpd
        disable     = yes
}
```

Exemple

```
# default : on
# description : The telnet server serves telnet sessions ; it uses \
#       unencrypted username/password pairs for authentication.
service telnet
{
        disable = no
        only-from = 193.52.237/24 193.52.236/24
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
}
```

Où en sommes nous ?

1. Lancement des services
2. Les supers démons inetd/xinetd
3. **Utilisation de SSH pour l'administration**

Génération de clés

Attention ce mécanisme suppose que la clé privée du serveur est gardée en sécurité. Cette méthode de connexion sera utilisé pour la synchronisation des répertoires utilisateur (web).

```
# génération des clés publiques et privées du serveur
# cette étape n'est à faire qu'une seule fois

ssh-keygent -t dsa

# ne pas entrer de passphrase seulement CR
# en résultat deux fichiers ont été générés :
# la clé privée
# /root/.ssh/id_dsa
# la cle publique
# /root/.ssh/id_dsa.pub

# copier la clé publique sur les machines distantes puis
# la concatener au fichier authorized_keys du répertoire .ssh

cat id_dsa.pub >> /root/.ssh/authorized_keys

# faire un test depuis le serveur
```

Tunnel ssh

Permet de faire transiter dans une communication cryptée tout protocole non sûre.

Les tunnels ssh s'appuient sur le renvoi de port (port forwarding)

- utilisation de l'option -L de n'importe quel client ssh
- diposer d'une connexion ssh sur un serveur
- créer le tunnel et se connecter sur un port local

Syntaxe générale :

```
ssh -L sortant :localhost :entrant
login@serveurSSH
```

Le protocole NTP

Il faut avant tout vérifier que le service ntpd est bien démarré : lancer l'utilitaire d'administration `setup` pour les machines sous RedHat et choisir `system services`. Le fichier `/etc/ntp.conf` doit contenir les lignes suivantes :

```
server ntp.univ-lyon1.fr
fudge  ntp.univ-lyon1.fr stratum 10
```

Il est possible de la configurer soit en editant directement le fichier `/etc/ntp.conf` ou en utilisant l'outil graphique `redhat-config-time`.

Le proxy cache SQUID

Squid est un serveur proxy cache qui supporte les protocoles HTTP, FTP et SSL :

- un proxy est un mandataire
- lorsqu'un passerelle joue également le rôle de proxy, cela signifie que les postes clients :
  - ne se connectent pas directement à Internet,
  - ils demandent au proxy de télécharger pour eux les pages dont ils ont besoin.
- il sera donc possible d'effectuer un filtrage sur les connexion lors de l'action du proxy (redirecteur)

Installation

Préparation :

```
# groupadd proxy
# useradd -g proxy proxy
# passwd proxy
# mkdir /var/log/squid  => stockage des logs
# mkdir /home/squid  => stockage de cache
# chwon proxy :proxy /var/log/squid /home/squid
```

Hypothèse réseau :

- Adressage de type : 192.168.0.x
- Adresse du serveur proxy : 192.168.0.1
- Nom du serveur proxy : serveur
- Port d'écoute de Squid : 3128.
- Postes client : 192.168.0.5 et plage 192.168.0.10 à 192.168.0.125

Installation

```
##### /etc/squid/squid.conf #####
...
http_port 192.168.0.1 :3128
...
cache_mem 100 MB
...
cache_dir ufs /var/spool/squid 5000 16 256
cache_access_log /var/log/squid/access.log
cache_log /var/log/squid/cache.log
cache_store_log /var/log/squid/store.log
...
# ACL : les acl permettent de spécifier les autorisations
# d'accès sur certains ports, et les adresses ip des stations
# l'ordre donné est important
acl all src 0.0.0.0/0.0.0.0
acl localhost src 127.0.0.1/255.255.255.255
...
acl serveur src 192.168.0.1
acl poste src 192.168.0.5
acl plageclients src 192.168.0.10-192.168.0.125
```

Installation

```
...
http_access allow serveur
...
http_access allow poste
http_access allow plageclients
# utilisateur et le groupe qui lance Squid :
cache_effective_user proxy
cache_effective_group proxy
...
# le nom de machine qui est serveur squid
visible_hostname serveur
...
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Il faut initialiser le cache à la première utilisation (structure de hachage) `/usr/sbin/squid -z`

Redirections

Deux cas :

- le proxy est la passerelle :

```
iptables -t nat -A PREROUTING -s 192.168.0.0/255.255.255.0
         -p tcp -m tcp --dport 80 -j REDIRECT --to-port 3128
```

- le proxy est une autre machine

```
iptables -t nat -A PREROUTING -s 192.168.0.0/255.255.255.0
         -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.1.2 :3128
```