Sécurité Linux – XinetD

I►LE SUPER DAEMON XINETD

eXtended INterNET Daemon est un Super-Démon qui s'intercale entre les services et les clients demandeurs. Mais à quoi sert il ?

Imaginons que la machine soit configurée en tant que Serveur **FTP**, le daemon **FTP** doit tourner en permanence pour surveiller son port et voir s'il y a des connexions… Cette manière de faire consomme des ressources et pourrait être beaucoup plus sécurisée …

XinetD tourne en permanence, on le démarre simplement en invoquant son script de démarrage sous **/etc/init.d/xinetd** .On pourra s'assurer de le configurer automatiquement dans les RunLevels concernés en manipulant les RCx.D ….

chkconfig –list xinetd

xinetd 0:off 1:off 2:on 3:on 4:on 5:on 6:off

Rappel : si l'on veut personnaliser les runlevels, il faut modifier le standard présent dans le script /etc/init.d/xinetd :

 # Default-Start: 2 3 4 5

Default-Stop: 0 1 6

Et ensuite on peut effectuer chkconfig –level 013456 off chkconfig –level 2 on

---

Le serveur(service) FTP est stoppé /etc/init.d/ssh stop

C'est XinetD qui surveille le port 21 (port du service FTP)

netstat -anlp | grep 21

tcp 0 0 0.0.0.0:**21** 0.0.0.0:* LISTEN **3466/xinetd**

Lorsqu'il y a une sollicitation FTP, XinetD va démarrer le service FTP et passer le relai.

Lorsque la connexion se termine, XinetD arrête le service FTP.
NB : *Les services sont enregistrés dans /etc/services (/etc/protocols).*

XinetD est en quelque sorte le **standardiste** de la machine, il reçoit des appels et les 'branche' sur les bons services …

Nous verrons plus tard que **XinetD** avec les **TCP_WRAPPERS** permet aussi de contrôler les connexions sur une machine au travers des fichiers **hosts.allow** et **hosts.deny.**

XinetD permet **en plus** de contrôler l'accès aux services … en modifiant certaines options d'accès :

- par les plages horaires
- par le Deny of Services (DoS),

 # NB : le DoS consiste à faire de multiples requêtes sur un serveur ce qui va accaparer toutes ses ressources jusqu'à le bloquer.

Le fichier de configuration de xinetd est **/etc/xinetd.conf :**

```
defaults
{

# Please note that you need a log_type line to be able to use
log_on_success
# and log_on_failure. The default is the following :
# log_type = SYSLOG daemon info

}
includedir /etc/xinetd.d
```

————————————————-

On créera un **répertoire /etc/xinetd.d** contenant autant de fichier qu'il y a de service, chaque fichier portant le nom du service et contenant :

```
service nom-du-service
{
socket_type = type de socket
wait = pour lancer un ou plusieurs instances du service
user = à qui appartiendra le service
server= chemin du serveur
server_args= argument du serveur (facultatif)
access_times=plage horaire (facultatif)
nice = priorité (facultatif)
}
```

Les options par service

id pour identifier un service, si on utilise pas ce paramètre, dans le fichier de log on aura le nom du service, avec id vous pouvez redéfinir le nom du service qui sera archivé dans le fichier de log (ftp-public au lieu simplement de ftp par exemple)

nice pour définir la priorité d'un service (-20 plus prioritaire 19 plus prioritaire)

port numéro du port associé au service défini dans /etc/services

protocol nom du protocole défini dans /etc/protocols

server le chemin du service

server_args arguments à passer au service

socket_type type de socket (stream pour TCP et dgram pour UDP), pour savoir si c'est l'un ou l'autre voir le fichier /etc/services

wait si wait=yes une seule instance du service peut être lancé à la fois, si wait=no, on lance un serveur à chaque nouvelle connexion (dans la limite du nombre d'instance max)

access_times pour définir une plage horaire la syntaxe est la suivante heure:minute-heure:minute, exemple pour une plage horaire entre 7h-12h et 14h-19h

access_times = 7:00-12:00 14:00-19:00

env pour passer une variable d'environnement avant de lancer le service

log_on_success = PID HOST USERID EXIT DURATION

#Va donner les lignes suivantes pour un telnet
#00/12/15@20:15:46: START: telnet pid=1889 from=192.168.13.10
#00/12/15@20:18:39: EXIT: telnet status=0 pid=1889 duration=173(sec)

#si problèmes de connexion
log_on_failure = HOST RECORD

#pour définir les clients autorisés à se connecter. Exemple avec le sous réseau 192.168.13.X
only_from = 192.168.13.0/24

#instances pour fixer le nombre maximum d'instances de serveur à lancer, ce qui permet d'éviter le DoS
instances = 60

#cps pour limiter le taux de connexion, il prend deux arguments, le premier fixe le nombre max de connexion à gérer par seconde, si le nombre de connexion dépasse ce chiffre, le service sera temporairement désactivé. Le deuxième argument fixe le nombre de secondes pendant lesquelles le service restera inactivé.
cps = 25 30
exemple avec log_on_success, si celui-ci est définit dans les paramètres par défaut avec les options suivantes PID HOST USERID EXIT DURATION, on veut avoir pour un service particulier uniquement PID et HOST on écrira donc :
log_on_success -= USERID EXIT DURATION

De même si on a log_on_success défini à PID HOST et qu'on veut rajouter DURATION on écrira :
log_on_success += DURATION

Exemple avec pop3

```
service pop3
{
socket_type = stream
wait = no
user = root
server = /usr/sbin/ipop3d
log_on_success += USERID
log_on_failure += USERID
}
```

Avec ftp avec une plage horaire et une limitation à 4 personnes connectées simultanément

```
service ftp
{
socket_type = stream
wait = no
protocol = tcp
user = root
server = /usr/sbin/in.ftpd
instances = 4
access_times = 7:00-12:00 14:00-17:00
nice=15
}
```

```
service xxx
{
disable = no
socket_type = stream
wait = no
user = root
server = /usr/sbin/sshd
```

```
server_args = -i
nice = 10
instances = 2
access_times = 7:00-12:00 14:00-17:00
}
```