

Cours 5 : Sécurité minimale sous Unix

Christophe Gonzales

31015 — Principes et pratiques de l'administration des systèmes

Principes fondamentaux

Principes

- Empêcher toute intrusion \implies impossible
- Rendre le travail nécessaire à l'intrusion $>$ gain qu'on en tire

Identifier l'« ennemi »

- Intrusion « locale » : accès physique à la machine
 - protéger le bios
 - protéger la séquence de démarrage
 - droits d'accès des fichiers (de configuration)
- Intrusion distante : accès par le réseau
 - réduire l'ouverture des ports
 - protéger les démons

Cours 5 : Sécurité minimale sous Unix

2/43

Protection locale (1/5)

- BIOS \implies démarrage de la machine sur CD-ROM
- install de linux à partir d'un CD-ROM \implies *root* :
 - ALT-F2 pendant l'install \implies console *root*
 - lancer le CD avec `linux rescue` \implies console *root*

contre-mesures

- BIOS : interdire le boot sur CD / disquette
- Interdire l'édition du bios \implies mot de passe

Protection locale (2/5)

- Grub : édition des lignes de démarrage :
 - 1 taper « e » sur une des lignes
 \implies accès aux options de démarrage
 \implies choix du runlevel d'init
 - 2 modification de ces paramètres :
`kernel /boot/vmlinuz ro root=/dev/sda6 1`
 - 3 appuyer sur entrée puis sur « b » pour booter
- Grub contient une interface avec ligne de commande (« c »)
`root (hd0,0)`
`cat /etc/passwd`
`cat /etc/shadow`

 \implies mot de passe pour accéder aux commandes de Grub

Cours 5 : Sécurité minimale sous Unix

3/43

Cours 5 : Sécurité minimale sous Unix

4/43

Protection locale : grub2 (3/5)

- /boot/grub/grub.cfg :
 - fichier de configuration exploité par grub au boot
 - fichier généré par update-grub
 - /etc/default/grub : configuration générale de grub (hors règles spécifiques aux OS)
 - /etc/grub.d/ : répertoire des scripts utilisés pour générer grub.cfg
- | | | | |
|-----------------|--------------|---------------|-----------|
| 00_header | 10_linux | 20_memtest86+ | 40_custom |
| 05_debian_theme | 20_linux_xen | 30_os-prober | 41_custom |

Protection locale : grub2 (4/5)

contre-mesure

- 1 ajouter un mot de passe dans /etc/default/grub :
 - 1 générer un mot de passe chiffré :

```
grub-mkpasswd-pbkdf2
Enter password : *****
Your PBKDF2 is grub.pbkdf2.sha512.1000.BCF6F9BC
```
 - 2 éditer /etc/grub.d/40_custom et rajouter les lignes :

```
cat <<EOF
set superusers="root"
password_pbkdf2 root grub.pbkdf2.sha512.1000.BCF6F9BC
EOF
```
- 2 éditer /etc/grub.d/10_linux et remplacer :

```
CLASS="--class gnu-linux --class gnu --class os" par
CLASS="--class gnu-linux --class gnu --class os
--users \"\""
```
- 3 exécuter update-grub
- 4 interdire en lecture /boot/grub/grub.cfg et /etc/default/grub à quiconque sauf root

Exemple de /boot/grub/grub.cfg (5/5)

```
### BEGIN /etc/grub.d/10_linux ###
menuentry 'Ubuntu, with Linux 2.6.35-31-generic'
--class ubuntu --class gnu-linux --class gnu
--class os --users "" {
  recordfail
  insmod part_msdos
  insmod ext2
  set root='(hd0,msdos1)'
  search --no-floppy --fs-uuid --set 1187ab9d-a203-4dde
  linux /vmlinuz-2.6.35-31-generic root=UUID=628d80b2
    ro quiet splash
  initrd /initrd.img-2.6.35-31-generic
}
```

Quelques pistes pour la sécurité locale (1/2)

- 1 droits d'accès des fichiers :
 - éviter les droits en lecture : /etc/grub.conf, /etc/shadow...
 - faire la chasse aux fichiers SUID/SGID :

```
find / -type f \( -perm -04000 -o -perm -02000 \)
```
 - aux fichiers en écriture pour tout le monde :

```
find / -perm -2 ! -type 1 -ls
```
 - aux fichiers sans propriétaire :

```
find / \( -nouser -o -nogroup \)
```
 - aux .rhosts :

```
find /home -name .rhosts -print
```
- 2 choix judicieux des passwords : utiliser *crack* et *John the ripper*
- 3 utiliser un système testant l'intégrité du système (ex : tripwire)

Quelques pistes pour la sécurité locale (2/2)

- 4 utiliser un emplacement sécurisé pour les logs :
(syslog peut envoyer les logs vers un serveur dédié)
- 5 utiliser les bonnes options dans /etc/fstab :
 - nosuid pour les partitions en écriture pour d'autres que root
 - nodev dans les home directories
 - nodev et noexec dans /var, /tmp...
- 6 update régulière des logiciels et packages
⇒ supprime les trous de sécurité
- 7 ne pas rajouter de chemin au PATH par défaut de root
commandes ⇒ taper tout le chemin à partir de la racine /

Sécurité distante

Accès internet et conséquences

Accès à internet ⇒ problèmes de sécurité

- chevaux de Troie
- *sniffer* : fuite de mots de passe
- *exploits* : trous de sécurité dans des applications
⇒ vulnérabilité de root
- *déni de service* : application ⇒ incapable de répondre
aux requêtes des utilisateurs (e.g., saturation)
-

Solution : limiter les accès de et vers internet

Limitation des connexions (1/2)

services exécutés au démarrage ⇒ la plupart des problèmes

```
root@msLDAP:~# systemctl --type=service
UNIT                                LOAD    ACTIVE SUB    DESCRIPTION
3i015.service                       loaded active running service de test 3i015
acpid.service                       loaded active running ACPI event daemon
anacron.service                     loaded active running Run anacron jobs
apache2.service                     loaded active running LSB: Apache2 web server
atd.service                         loaded active running Deferred execution scheduler
avahi-daemon.service                loaded active running Avahi mDNS/DNS-SD Stack
console-setup.service               loaded active exited LSB: Set console font and ke
cron.service                        loaded active running Regular background program p
dbus.service                        loaded active running D-Bus System Message Bus
getty@tty1.service                  loaded active running Getty on tty1
hddtemp.service                     loaded active exited LSB: disk temperature monito
kbd.service                         loaded active exited LSB: Prepare console
keyboard-setup.service              loaded active exited LSB: Set preliminary keymap
lightdm.service                     loaded active running Light Display Manager
```

- limiter les accès vers les services avant connexion :
 - utilisation des iptables
 - utilisation de xinet et des tcp wrappers
- limiter les accès vers les services après connexion :
 - utilisation des *pluggable authentication modules* (PAM)
 - fichiers de configuration

IP tables

IP table = filtre de paquets
= examine les en-têtes des paquets transitant par une machine et décide quoi faire avec (accepter le paquet, le supprimer, etc)

- 1994 : 1ère génération *ipfw* porté de BSD à linux
- 1998 : 2ème génération *ipchains*
- 1999 : 3ème génération *iptables*
- 2014 : 4ème génération *nftables* – encore en développement

IP table \implies module du noyau linux

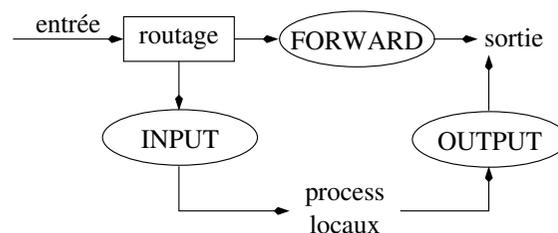
Principe des iptables

IP tables = ensemble de règles indiquant le sort des paquets

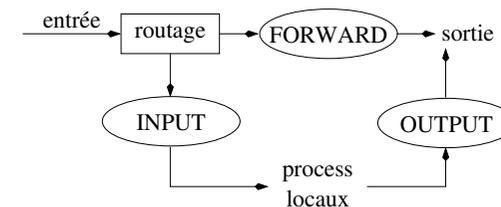
- Organisation hiérarchique :



- Tables : **filter**, nat, mangle, etc.
- 3 chaînes par défaut : **INPUT**, **OUTPUT** et **FORWARD**



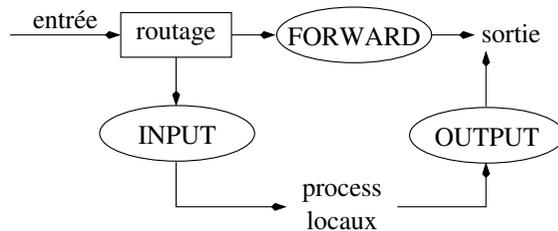
Traversée des filtres



- 1 paquet arrive \implies le noyau regarde sa destination (*routage*)
- 2 Si paquet destiné à la machine \implies transmis à la chaîne **INPUT**
- 3 Sinon :
 - si noyau linux non configuré en mode *forwarding* ou si on ne sait comment faire le forward : alors détruire le paquet
 - sinon si paquet destiné à une autre interface réseau : alors transmettre le paquet à la chaîne **FORWARD**. Si celle-ci l'accepte, envoyer le paquet en sortie
- 4 1 logiciel tournant sur la machine peut envoyer des paquets transmettre ceux-ci à la chaîne **OUTPUT**

Principe des iptables

- Le noyau démarre avec 3 listes (*chaînes*) : **INPUT**, **OUTPUT** et **FORWARD**



- paquet arrive sur chaîne \Rightarrow celle-ci décide de son sort :
 - DROP** : le paquet est détruit
 - ACCEPT** : le paquet continue sa traversée dans le diagramme
- chaîne = liste de règles :
 - règle = si en-tête du paquet a telle forme alors faire...
 - si une règle ne matche pas, on passe à la suivante
 - s'il n'y a plus de règles, on applique la *politique* de la chaîne

Exemple d'utilisation (1/6)

- Configuration du parc informatique :



- Fichier de configuration : mis où on veut (par exemple /etc/iptables)
- Iptables : application de règles / remise à zéro :

```
iptables-restore < /etc/iptables
iptables -F
```
- Voir la configuration des iptables : `iptables -L`

Ici, on va configurer la passerelle

Anatomie d'une iptable

- Contenu des iptables sur vos machines :

```
*filter
:INPUT ACCEPT
:FORWARD ACCEPT
:OUTPUT ACCEPT
COMMIT
```

⚠ Il y a aussi une table `*nat`, dont je ne parlerai pas ici !

- `*filter` : table concernée par les prochaines règles (possibilité d'autres tables `*mangle`, *etc.*)
- `:INPUT ACCEPT` : politique par défaut de la chaîne INPUT
- `COMMIT` : obligatoire, applique les règles de la table

Exemple d'utilisation (2/6)



- Remise à zéro des iptables : on part d'un fichier vide
- Faire pointer les politiques des chaînes sur **DROP** :

```
*filter
:INPUT DROP
:OUTPUT DROP
:FORWARD DROP
COMMIT
```

\Rightarrow aucun paquet ne passe plus nulle part :

- pings réseau privé \rightarrow passerelle (INPUT) ✗
- pings passerelle \rightarrow réseau privé (OUTPUT) ✗
- pings réseau privé \rightarrow internet (FORWARD) ✗

Exemple d'utilisation (3/6)



- notre machine = sûre \Rightarrow process communiquent via le loopback
 - A INPUT -i lo -j ACCEPT
 - A OUTPUT -o lo -j ACCEPT

| action | réponse | cause |
|--|---------|---|
| ping passerelle \rightarrow 127.0.0.1 | ✓ | INPUT : lo = ACCEPT OUTPUT : lo = ACCEPT |
| ping réseau privé \rightarrow passerelle | ✗ | INPUT : DROP par défaut |
| ping internet \rightarrow passerelle | ✗ | INPUT : DROP par défaut |
| ping passerelle \rightarrow réseau privé | ✗ | OUTPUT : DROP par défaut |
| ping passerelle \rightarrow internet | ✗ | OUTPUT : DROP par défaut |
| ping réseau privé \rightarrow internet | ✗ | FORWARD : DROP par défaut |

communications \Rightarrow règles

ping machine A \rightarrow machine B

- A envoie un paquet ICMP vers B
 \Rightarrow chaîne **OUTPUT** de A
- B reçoit le paquet
 \Rightarrow chaîne **INPUT** de B
- B renvoie la réponse (nouveau paquet ICMP) vers A
 \Rightarrow chaîne **OUTPUT** de B
- A reçoit la réponse de B
 \Rightarrow chaîne **INPUT** de A

pour que A et B communiquent, leurs deux chaînes **INPUT** et **OUTPUT** doivent accepter les paquets

Exemple d'utilisation (4/6)



- le réseau local est sûr :
 - A INPUT -i enp0s3 -j ACCEPT
 - A OUTPUT -o enp0s3 -j ACCEPT

| action | réponse | cause |
|--|---------|---|
| ping passerelle \rightarrow 127.0.0.1 | ✓ | INPUT : lo = ACCEPT OUTPUT : lo = ACCEPT |
| ping réseau privé \rightarrow passerelle | ✓ | INPUT : enp0s3 = ACCEPT |
| ping passerelle \rightarrow réseau privé | ✓ | OUTPUT : enp0s3 = ACCEPT |
| ping internet \rightarrow passerelle | ✗ | INPUT : DROP par défaut |
| ping passerelle \rightarrow internet | ✗ | OUTPUT : DROP par défaut |
| ping réseau privé \rightarrow internet | ✗ | FORWARD : DROP par défaut |

État actuel de /etc/iptables

```
*filter
:INPUT DROP
:FORWARD DROP
:OUTPUT DROP
-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A INPUT -i enp0s3 -j ACCEPT
-A OUTPUT -o enp0s3 -j ACCEPT
COMMIT
```

| action | réponse | cause |
|--|---------|---|
| ping passerelle \rightarrow 127.0.0.1 | ✓ | INPUT : lo = ACCEPT OUTPUT : lo = ACCEPT |
| ping réseau privé \rightarrow passerelle | ✓ | INPUT : enp0s3 = ACCEPT OUTPUT : enp0s3 = ACCEPT |
| ping passerelle \rightarrow réseau privé | ✓ | OUTPUT : enp0s3 = ACCEPT INPUT : enp0s3 = ACCEPT |
| ping internet \rightarrow passerelle | ✗ | INPUT : DROP par défaut |
| ping passerelle \rightarrow internet | ✗ | OUTPUT : DROP par défaut |
| ping réseau privé \rightarrow internet | ✗ | FORWARD : DROP par défaut |

Exemple d'utilisation (5/6)



- accepter les connexions réseau privé → internet :

```
-A FORWARD -i enp0s3 -o enp0s8 -m conntrack --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT
```
- internet → réseau privé : accepter seulement les connexions déjà établies ou en connexion avec elles :

```
-A FORWARD -i enp0s8 -o enp0s3 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Exemple d'utilisation (6/6)

Installation d'un serveur DNS sur la passerelle
⇒ devra pouvoir envoyer ses requêtes sur internet :

- Votre serveur : envoie requêtes UDP → port 53 d'un serveur DNS
- attend réponses venant d'un port 53 → port > 1024

⇒ Rajouter les règles suivantes :

```
-A OUTPUT -o enp0s8 -p udp --dport 53 -j ACCEPT
-A INPUT -i enp0s8 -p udp --sport 53 -j ACCEPT
```

On peut aussi spécifier les ports sources :

```
-A OUTPUT -o enp0s8 -p udp --sport 1024:
--dport 53 -j ACCEPT
-A INPUT -i enp0s8 -p udp --dport 1024:
--sport 53 -j ACCEPT
```

Paramètres usuels à utiliser dans les règles

| param. | signification | exemple |
|---------|---|--------------|
| -i | interface réseau d'entrée du paquet | lo,enp0s3 |
| -o | interface réseau de sortie du paquet | lo,enp0s3 |
| -p | protocole du paquet | tcp,udp,icmp |
| -s | adresse IP de la source du paquet | 192.168.1.2 |
| --sport | numéro de port source du paquet | 80 |
| -d | adresse IP de la destination du paquet | 192.168.1.2 |
| --dport | numéro de port de destination du paquet | 80 |



Choisir seulement les paramètres utiles !

Exemple : `-A INPUT -p tcp -s 192.168.1.1 --sport 389 -j ACCEPT`

Comment connaître les ports utilisés ?

- Fichier `/etc/services` :

```
ftp-data 20/tcp
ftp      21/tcp
ssh     22/tcp          # SSH Remote Login Protocol
ssh     22/udp
telnet  23/tcp
smtp    25/tcp          mail
domain  53/tcp          # Domain Name Server
domain  53/udp
http    80/tcp          www          # WorldWideWeb HTTP
sunrpc  111/tcp         rpcbind     # RPC 4.0 portmapper TCP
sunrpc  111/udp         rpcbind     # RPC 4.0 portmapper UDP
nfs     2049/tcp        # Network File System
nfs     2049/udp        # Network File System
amanda  10080/tcp       # amanda backup services
amanda  10080/udp
```

Comment connaître les ports utilisés ?

- Pour les services s'appuyant sur RPC :

```
[root@msLDAP yp]# rpcinfo -p
program  vers  proto  port  service
100000   4     tcp    111   portmapper
100000   3     tcp    111   portmapper
100000   4     udp    111   portmapper
100000   3     udp    111   portmapper
100011   1     udp    875   rquotad
100011   1     tcp    875   rquotad
100021   4     udp    32769 nlockmgr
100021   4     tcp    32803 nlockmgr
100003   4     tcp    2049  nfs
100003   4     udp    2049  nfs
100005   3     udp    892   mountd
100005   3     tcp    892   mountd
100004   2     udp    8343  ypserv
100004   2     tcp    8343  ypserv
600100069 1     udp    8353  fypxfrd
600100069 1     tcp    8353  fypxfrd
```

Comment connaître les ports utilisés ?

- Rajouter des instructions de log dans les iptables :

```
# logs
-A INPUT -s 192.168.X.Y -j LOG
-A OUTPUT -d 192.168.X.Y -j LOG
```

⇒ Fichier /var/log/kern.log :

```
Jan 25 13:10:29 msLDAP kernel: [ 7521.487541]
IN=enp0s3 OUT= MAC=08:00:27:59:2e:c8:08:00:27:86:5b:51:08:00
SRC=192.168.1.2 DST=192.168.1.1 LEN=60 TOS=0x00 PREC=0x00
TTL=64 ID=25888 DF PROTO=TCP SPT=943 DPT=2049 WINDOW=5840
RES=0x00 SYN URGP=0

Jan 25 13:10:32 msLDAP kernel: [ 7524.493154]
IN= OUT=enp0s3 MAC=08:00:27:59:2e:c8:08:00:27:86:5b:51:08:00
SRC=192.168.1.1 DST=192.168.1.2 LEN=60 TOS=0x00 PREC=0x00
TTL=64 ID=25889 DF PROTO=TCP SPT=2049 DPT=833 WINDOW=5840
RES=0x00 SYN URGP=0
```

Xinetd et les tcp wrappers

2 moyens d'exécuter les services :

- les lancer à la main (les scripts `systemctl...`)
- les lancer quand un client les demande ⇒ xinetd

`/usr/sbin/xinetd` : le lanceur de services

fonctionnement de xinetd

- Attend et prend en compte des requêtes pour des services
- Pour certaines, les sous-traite (création de processus, activation de serveurs : `/usr/sbin/in.ftpd`, *etc.*)
- Services en charge définis dans : `/etc/xinetd.conf`
- Arguments transmissibles lors de l'activation du serveur
- « Effective user » précisé pour l'activation du serveur

Configuration de xinetd (1/4)

```
$ more /etc/xinetd.conf
# Simple configuration file for xinetd
# Some defaults, and include /etc/xinetd.d/
defaults
{
    instances          = 60
    log_type           = SYSLOG authpriv
    log_on_success     = HOST PID
    log_on_failure     = HOST
    cps                = 25 30
}
includedir /etc/xinetd.d

$ ls -l /etc/xinetd.d
total 88
-rw-r--r--  1 root  root   295 Jul  5  2002 daytime
-rw-r--r--  1 root  root   315 Jul  5  2002 daytime-udp
-rw-r--r--  1 root  root   287 Jul  5  2002 echo
-rw-r--r--  1 root  root   306 Jul  5  2002 echo-udp
-rw-r--r--  1 root  root   317 Jul  5  2002 finger
-rw-r--r--  1 root  root   359 Jul  5  2002 rexec
-rw-r--r--  1 root  root   376 Jul  5  2002 rlogin
-rw-r--r--  1 root  root   478 Jul  5  2002 rsh
-rw-r--r--  1 root  root   317 Jul  5  2002 rsync
```

Configuration de xinetd (2/4)

```
$ more /etc/xinetd.d/*      # extraits

/etc/xinetd.d/daytime :

# default: off
# description: A daytime server. This is the tcp version.
service daytime
{
    disable          = yes
    type             = INTERNAL
    id               = daytime-stream
    socket_type      = stream
    protocol         = tcp
    user             = root
    wait             = no
}
```

Configuration de xinetd (3/4)

```
/etc/xinetd.d/finger :

# default: on
# description: The finger server answers finger requests.
# Finger is a protocol that allows remote users
# to see information such as login name and last login
# time for local users.
service finger
{
    disable          = yes
    socket_type      = stream
    wait            = no
    user            = nobody
    server          = /usr/sbin/in.fingerd
}
```

Configuration de xinetd (4/4)

```
/etc/xinetd.d/rsync
# default: off
# description: The rsync server is a good addition to an
# ftp server, as it allows crc checksumming etc.
service rsync
{
    disable          = yes
    socket_type      = stream
    wait            = no
    user            = root
    server          = /usr/bin/rsync
    server_args      = --daemon
    log_on_failure   += USERID
}
```

Intérêt de xinetd : l'utilisation du démon tcpd

Le démon /usr/sbin/tcpd (TCP wrapper)

- Historiquement, en complément de inetd, contrôle d'accès aux services réseaux
- règles d'accès dans :
/etc/hosts.allow et /etc/hosts.deny
Cf. < man tcpd >
 - < man 5 hosts.allow >
 - < man 5 hosts.deny >
 - < man portmap >

Le démon /usr/sbin/tcpd (2/2)

Syntaxe des fichiers :

- service : host [: Shell_command]
 - "service" = un ou plusieurs noms ou les mots clés **ALL**, **ALL EXCEPT?**
 - "host" = un ou plusieurs noms ou adresses IP d'hôte ou de réseau ou un nom de fichier (/xxx) contenant ces éléments, ou les mots clés **ALL**, **LOCAL** (noms d'hôtes sans '.'), **UNKNOWN**, **PARANOID** (échec de la recherche d'adresse IP), **ALL EXCEPT?**, **ALL EXCEPT LOCAL**
 - "commande" = actions à effectuer
- 1 lecture fichier hosts.allow ⇒ potentiel succès
 - 2 sinon lecture fichier hosts.deny ⇒ potentiel échec
 - 3 Sinon succès

succès ⇒ le serveur est activé

écriture possible dans des fichiers de log

Le fichier hosts.allow

```
$ more /etc/hosts.allow
# hosts.allow This file describes the names of the hosts
# which are allowed to use the local INET services,
# as decided by the '/usr/sbin/tcpd' server.

portmap : 172.17.116.0/255.255.255.0, \
          172.27.94.128/255.255.255.128 : ALLOW

in.telnetd : 132.227.60.30 : ALLOW
sshd : ALL : ALLOW
in.ftpd : LOCAL : ALLOW
in.rshd : ALL except 132.227.203.45 : ALLOW
```

Le fichier hosts.deny

```
$ more /etc/hosts.deny
# hosts.deny This file describes the names of the hosts
# which are *not* allowed to use the local INET services,
# as decided by the '/usr/sbin/tcpd' server. The portmap
# line is redundant, but it is left to remind you that the
# new secure portmap uses hosts.deny and hosts.allow.
# In particular you should know that NFS uses portmap!

in.telnetd : ALL : spawn (/usr/sbin/safe_finger -l @%h | \
                          /bin/mail -s %d-%h root)& : DENY
in.ftpd : ALL : spawn (/usr/sbin/safe_finger -l @%h | \
                       /bin/mail -s %d-%h root)& : DENY

ALL:ALL@ALL
```

Quelques pistes pour la sécurité distante

- supprimer tous les services non strictement nécessaires
- supprimer les r-commandes (rlogin, rexec, rsh) : les mots de passe circulent en clair sur le réseau
- supprimer les .rhosts et /etc/hosts.equiv
- Faire du monitoring : vérifier les logs (logs manquants, incomplets ou avec des droits d'accès étranges, redémarrages de la machine ou de services, entrées de su ou de logins curieux, ...)
Les logs se trouvent dans /var/log
- exécuter régulièrement des logiciels testant la fiabilité de votre machine : nagios, ganglia, nessus, etc.
- bien éditer les fichiers de configuration

Exemple de fichier de configuration

```
$cat /etc/ssh/sshd_config
Port 22
Protocol 2

# Logging
# obsolete QuietMode and FascistLogging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6

# To disable tunneled clear text passwords, change to no here !
#PermitEmptyPasswords no
PasswordAuthentication yes

AllowTcpForwarding yes
GatewayPorts no
X11Forwarding yes
#UseLogin no
#UsePrivilegeSeparation yes
```

Quelques lectures...

L'administration système en général pour Unix :

- Jean-Michel Moreno (2003) « Unix administration : Systèmes et réseaux », 3ème édition, Dunod.
- <http://casteyde.christian.free.fr/system/linux/guide/online/book1.html>

La sécurité sous Linux en général :

- <http://www.linuxsecurity.com/>
- http://www.linuxtopia.org/online_books/linux_administrators_security_guide/

PAM pour Linux :

- <http://www.kernel.org/pub/linux/libs/pam/>