Kernel Linux

notes Kernel Linux

Table des matières

1. Modules

- obtenir les arguments acceptés par un module du noyau
```
>modinfo $nom_module
```
les arguments sont passés au moment du chargement du module grâce à la commande options dans le fichier /etc/modules.conf
```
|options snd cards-limit=2
```
- empêcher le chargement d'un module par modprobe insérer dans /etc/modules.conf
```
|install $nom_module false
```

2. initrd

C'est un système de montage de disque ram, qui intervient juste après le chargement du noyau, et qui permet de charger des modules critiques et s'affranchir ainsi d'avoir à les compiler directement dans le noyau. On charge par exemple le driver de la carte scsi avant de monter le système de fichier racine qui se trouve sur un disque scsi. Le fichier /boot/initrd.img est en fait une image compressée d'un système de fichier ext2 racine contenant les modules critiques. Un script (mkinitrd) permet de créer automatiquement ce système de fichier en spécifiant les noms des modules critiques. Ce fichier est pris en compte par le chargeur de noyau (lilo ou grub), grace à une ligne une option dans le fichier de configuration (/etc/lilo.conf par exemple)

```
|image=/boot/vmlinuz-2.4.19-13mdk
|       ...
|       initrd=/boot/initrd-2.4.19-13mdk.img
|       ..
```

Certaines informations de /etc/modules.conf sont significatives pour mkinitrd pour aller chercher des noms de modules critiques directement. Par exemple tous les modules apparaissant à droite de probeall scsi_hostadapter sont automatiquement mis dans l'image.

2.1. Exemples pratiques

1. Mettre des périphériques IDE sur une chaine SCSI virtuelle pour pouvoir utiliser cdrecord par exemple. Le module s'appelle ide-scsi et doit apparaitre sur la ligne
```
probeall scsi_hostadapter
>mkinitrd /boot/initrd-2.4.19-13mdk.img 2.4.19-13mdk
```
2. Charger le gestionnaire de frame buffer pour radeon au demarrage (desactiver vesa)
```
>mkinitrd -f --preload "fbcon-cfb32 fbcon-cfb24 fbcon-cfb16 fbcon-cfb8
radeonfb" initrd-2.4.20-2mdkcustom.img 2.4.20-2mdkcustom
```

Note:

L'argument du noyau video=vesa:option permet d'activer des options supplémentaires pour vesa (Voir /usr/src/linux/Documentation/fb/vesafb.txt), et video=radeon permet de sélectionner le bon pilote sans avoir a desactiver vesa

3. Bootsplash

Attache une image jpg dans une image initrd et permet le boot en mode graphique si le framebuffer et bootsplash sont supportés par le noyau et activés. les scripts utiles sont dans /usr/share/bootsplash/scripts.

Le script make-boot-splash se sert de l'entrée vga= du fichier /etc/modules.conf pour connaitre la résolution de l'image à insérer dans l'initrd (Voir /usr/src/linux/Documentation/fb/vesafb.txt pour les modes). On peux aussi spécifier la résolution en 2nd argument (ex. 1024 eq 1024x768) Pour que l'image apparaisse, il faut que le frame buffer soit en 16 bit car c'est /usr/src/linux/drivers/video/fbcon-cfb16.c qui est concerné (patché) par bootsplash.

4. Compilation du noyau

4.1. Configuration

Il faut disposer des sources, et configurer les options souhaitées dans le noyau. L'emplacement standard des sources du noyau est /usr/src/linux. La configuration se fait de manière interactive sous X-Windows

```
>make xconfig
```

et en mode texte

```
>make menuconfig
```

Le fichier contenant toutes les options selectionnées est nommé .config. On peut reprendre une config existante et ne configurer que les nouvelles options proposées lors d'une mise à jour d'un noyau avec

```
>make oldconfig
```

4.2. Compilation

1. Il peut être necessaire de nettoyer l'arborescence des fichiers objets déjà compilé au cas ou l'on a changé radicalement la configuration du noyau:
   ```
   >make clean
   ```
   ou encore mieux
   ```
   >make mrproper
   ```
2. Ensuite générer les dépendances
   ```
   >make dep
   ```
3. Finalement générer l'image du noyau, les modules, et installer les modules (/lib/modules...)
   ```
   >make bzimage
   ```

```
>make modules
>make modules_install
```

4.3. Génération de RPMs

Pour faciliter la mise à jour de plusieurs machines ou pour simplement pour garder sa base RPM synchronisé lors de la mise à jour d'un noyau ou l'installation d'un nouveau, il est préférable de générer des paquetages rpms.

1. Récupérer un RPM source: kernel.src.rpm
   ```
   >rpm --install kernel.src.rpm
   ```
2. Éditer le fichier /usr/src/RPMS/SPECS/kernel.spec par exemple pour ne pas générer le noyau secure, SMP, ou enterprise
   ```
   >rpm -ba kernel.spec
   ```
3. On peut aussi ne générer que les modules voulus en compliquant un peu la procédure
   1. Passer la phase de préparation (prep) du RPM; desarchive, applique les patches et génère les .configs
      ```
      >rpm -bp kernel.spec
      ```
   2. Convertir si necessaire le .config (version antérieure du noyau) a l'aide du makefile
      ```
      >make oldconfig
      ```
   3. Ecraser le .config dans /usr/src/RPMS/BUILD/kernel/arch/i386/defconfig
   4. Compiler
      ```
      >rpm -bc --short-circuit kernel.spec
      ```
   5. Installer dans le répertoire temporaire
      ```
      >rpm -bi --short-circuit kernel.spec
      ```
   6. Générer les paquetages binaires
      ```
      >rpm -bb --short-circuit kernel.spec
      ```

5. ACPI

ACPI est une norme de gestion de l'energie visant a remplacer APM et supportée par la pluspart des matériels récent. ACPI est géré par le noyau. La table DSDT est une microcode qui décrit les zones, les adresse mémoires conrespondant à la récupération des infos ACPI et l'activation des fonctionalités du matériel à laquelle elle est liée. ACPI étant un standard relativement récent il est souvent necessaire de remplacer cette table au niveau du noyau pour compenser des bugs. (http://www.cpqlinux.com/acpi-howto.html)

- Paramètres noyau Desactiver ACPI
  ```
  |acpi=off
  ```
  Desactiver les APIC (interruptions programmables), qui peuvent provoquer un gel lors de l'amorçage
  ```
  |noapic
  ```
- Activer tous les modules ACPI
- Remplacement de la table DSDT
  ```

  >pacpidump
  ```

```
>dsdt.asl
```
éditer dsdt.asl
```
>iasl dsdt.asl
>cp dsdt.hex /usr/sr/linux/drivers/acpi/dsdt.hex
```
- Charger les modules suivant le besoin editer /etc/modules.conf
```
|alias acpi ac battery button procesor thermal
>modprobe acpi
```
Les informations sont accessibles dans /proc/acpi
- Arrêt sur disque: Permet de sauver une image de la mémoire sur le disque avant d'éteindre la machine. A l'allumage, l'ordinateur amorçe assez rapidement, et revient à l'état précédant l'arrêt L'image s'écrit dans la partition swap. Il faut préciser dans les paramètres du noyau quelle est la partition utilisée. éditer /etc/lilo.conf
```
|image=...
|append=... resume=/dev/hda7
```
prévoir une entrée dans lilo sans resume au cas ça se passe mal. On réinitialise le swap par
```
>mkswap /dev/hda7
```

6. lilo

- changer la résolution fichier /boot/message-graphic
  E0 suivit d'une combinaison (XOR):
  - 0X80: mode (2 octets) exp: 0x101
  - 0x40
  - 0x20
  - 0x10
  - 0x08: menu (14 octects): timery(2), timerx(2), backcol(1), forcol(1), entryy(2), entryx(2), backcol(1), forcol(1), row(1), carlen(1)
  - 0x04
  - 0x02: putrec (4 octets): width(2), height(2)
  - 0x01

  exemple: 0X8A (16 octets): mode(2) + menu(14) + putrec(4)

  Pour changer la résolution

  1. changer les octects 3 et 4 (mode)
  2. changer les octects 19 20 (width) 21 22 (height)

7. Routage

- `/etc/resolv.conf` contient les serveurs de noms de domaine qui seront intérrogés

(entrées `nameserver`) ainsi que les noms de domaines automatiquement rajouté en suffixe des noms non qualifiés ().

- `/etc/nssswitch.conf` contient l'ordre des protocoles utilisé pour les requêtes de noms en fonction des services (networks, hosts, ...).
- tmdns est un service de dns multicast: un nom dans le domaine `local` est recherché en envoyant une requête dns sur l'adresse ip multicast `224.0.0.251`. Pour cela le fichier `/etc/resolv.conf` commence par

```
nameserver 127.0.0.1
search local
```

- multicast: pour un noyau qui supporte le multicast, il faut ajouter une route statique manuellement:

```
>route add -net 224.0.0.0 netmask 240.0.0 dev eth0
```

ou dans le fichier `/etc/sysconfig/static-routes`

- routed ou zebra sont des services de participation aux réseaux de mise à jour de tables de routage (protocole RIP). Les options de routed se règlent dans `/etc/sysconfig/routed`:

```
EXPORT_GATEWAY="true"
 SILENT="false"
```

8. IpTables

- ports peer to peer (edonkey, ...)
  - tcp: 1214,2234,4444,4661:4662,6346,6699,6882,9999
  - udp: 4665:4666
- masquerading

```
>iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
>echo 1 >/proc/sys/net/ipv4/ip_forward
```

- firewall bash (manuel)

```
#!/bin/bash
################################################################################
# IPTABLES VERSION
# This sample configuration is for a single host firewall configuration
# with no services supported by the firewall machine itself.
################################################################################

# USER CONFIGURABLE SECTION

# The name and location of the ipchains utility.
IPTABLES=iptables

# Our internal network address space and its supporting network device.
#OURNET="10.8.9.225/32"
OURNET="10.8.0.0/16"
OURBCAST="10.8.9.255"
OURDEV="eth1"

# The outside address and the network device that supports it.
```

```
ANYADDR="0/0"
ANYDEV="eth0"

# Services locaux du routeur
# router = RIP
# nameserver = WINS
LTCPIN="ssh"
LUDPIN="ssh,router"
LUDPCOMIN="ssh,netbios-ns,netbios-ssn,netbios-dgm,domain,nameserver,router"

# Hotes partageant des fichiers par samba entre les 2 réseaux
SMBHOSTIN="svr_bo svr_tras tresor01 svr_dgcpt"
SMBHOSTOUT="10.6.8.3 10.6.11.2"

# The TCP services we wish to allow to pass - "" empty means all ports
# oracle = 1521,1526
#
TCPIN="ssh,domain,nameserver,netbios-ns,netbios-ssn"
if [ $# -eq 1 -a "$1" = "-p" ]; then
TCPOUT="ssh,domain,nameserver,netbios-ns,netbios-ssn,1521,1526,squid,pop3,smtp"
else
TCPOUT="ssh,domain,nameserver,netbios-ns,netbios-ssn,1521,1526,pop3,smtp"
fi

# The UDP services we wish to allow to pass - "" empty means all ports
# note: comma separated
UDPIN="ssh,domain,nameserver,netbios-ns,netbios-ssn"
UDPOUT="ssh,domain,nameserver,netbios-ns,netbios-ssn,1521,1526"

# The ICMP services we wish to allow to pass - "" empty means all types
# ref: /usr/include/netinet/ip_icmp.h for type numbers
# note: comma separated
ICMPIN="0 3 8 11"
ICMPOUT="0 3 8 11"

# Logging; uncomment the following line to enable logging of datagrams
# that are blocked by the firewall.
LOGGING=1
if [ $LOGGING -eq 1 ]; then
 DROPMODE=LOGDROP
else
 DROPMODE=DROP
fi

NOTLOGUED="snmp,bootpc,cmip-man,164,snmp,1035"

# END USER CONFIGURABLE SECTION
################################################################################
# Flush tables
$IPTABLES -t filter -F
$IPTABLES -t nat -F
$IPTABLES -t mangle -F
# Efface les chaines utilisateurs
$IPTABLES -X
```

```
# Creer une chaine qui log selectivement et drop en même temps
$IPTABLES -N LOGDROP
# regles pour dropper sans trace
$IPTABLES -A LOGDROP -p icmp --icmp-type 9 -j DROP
$IPTABLES -A LOGDROP -p 9 -j DROP
$IPTABLES -A LOGDROP -m multiport -p udp --sports $NOTLOGUED -j DROP
$IPTABLES -A LOGDROP -m multiport -p udp --dports snmp -j DROP
# ce qui est passé est inscrit dans le log
$IPTABLES -A LOGDROP -j LOG
$IPTABLES -A LOGDROP -j DROP

# We want to deny incoming access by default.
$IPTABLES -P FORWARD DROP

# Filtre ce que l'on recoit de l'interface $ANYDEV
$IPTABLES -A INPUT -m multiport -p tcp -i $ANYDEV --dports $LTCPIN -j
ACCEPT
$IPTABLES -A INPUT -p icmp -i $ANYDEV -j ACCEPT
# Accepter tout ce qui n'est pas une demande de connexion
$IPTABLES -A INPUT -p tcp -i $ANYDEV ! --syn -j ACCEPT
$IPTABLES -A INPUT -m multiport -p udp -i $ANYDEV --dports $LUDPIN -j
ACCEPT
$IPTABLES -A INPUT -m multiport -p udp -i $ANYDEV --sports $LUDPCOMIN -j
ACCEPT
# Drop all datagrams destined for this host received from outside.
$IPTABLES -A INPUT -i $ANYDEV -j $DROPMODE

# SPOOFING
# We should not accept any datagrams with a source address matching ours
# from the outside, so we deny them.
$IPTABLES -A FORWARD -s $OURNET -i $ANYDEV -j $DROPMODE

# SMURF
# Disallow ICMP to our broadcast address to prevent "Smurf" style
attack.
$IPTABLES -A FORWARD -p icmp -i $ANYDEV -d $OURBCAST -j $DROPMODE

# We should accept fragments, in iptables we must do this explicitly.
$IPTABLES -A FORWARD -f -j ACCEPT


# TCP
# Accepter toutes les connexions TCP entre hôtes du trésor via
l'interface trésor
$IPTABLES -A FORWARD -p tcp -i $OURDEV -s $OURNET -d $OURNET -j ACCEPT

# We will accept all TCP datagrams belonging to an existing connection
# (i.e. having the ACK bit set) for the TCP ports we're allowing
through.
# This should catch more than 95 % of all valid TCP packets.
$IPTABLES -A FORWARD -m multiport -p tcp -d $OURNET --sports $TCPOUT !
--syn -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p tcp -s $OURNET --sports $TCPIN !
```

```
--syn -j ACCEPT

# TCP - INCOMING CONNECTIONS
# We will accept TCP datagrams from the outside only on the allowed TCP
ports.
$IPTABLES -A FORWARD -m multiport -p tcp -i $ANYDEV -d $OURNET --dports
$TCPIN -j ACCEPT
# Accepter les packets smb seulement vers les serveurs autorisés
for host_i in $SMBHOSTIN; do
 $IPTABLES -A FORWARD -m multiport -p tcp -i $ANYDEV -d $host_i --dports
netbios-dgm -j ACCEPT
done

# TCP - OUTGOING CONNECTIONS
# We will accept TCP datagrams on the allowed TCP ports.
$IPTABLES -A FORWARD -m multiport -p tcp -i $OURDEV -s $OURNET --dports
$TCPOUT -j ACCEPT
# Accepter les packets smb seulement vers les serveurs autorisés
for host_i in $SMBHOSTOUT; do
 $IPTABLES -A FORWARD -m multiport -p tcp -i $OURDEV -d $host_i --dports
netbios-dgm -j ACCEPT
done


# UDP
# Accepter toutes les connexions TCP entre hôtes du trésor via
l'interface tresor
$IPTABLES -A FORWARD -p udp -i $OURDEV -s $OURNET -d $OURNET -j ACCEPT

# UDP - INCOMING
# Accepter les paquets venants de ports identifiés vers n'importe quel
port
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $OURNET --sports
$UDPOUT -j ACCEPT
# Accepter les paquets venants de n'importe quel port vers des ports
servis
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $OURNET --dports
$UDPIN -j ACCEPT
# Accepter les packets smb seulement vers les serveurs autorisés
for host_i in $SMBHOSTIN; do
 $IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $host_i --dports
netbios-dgm -j ACCEPT
done

# UDP - OUTGOING
# Accepter les paquets venants de ports identifiés vers n'importe quel
port
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -s $OURNET --sports
$UDPIN -j ACCEPT
# Accepter les paquets partants de n'importe quels ports vers des ports
réputés servis
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -s $OURNET --dports
$UDPOUT -j ACCEPT
# Accepter les packets smb seulement vers les serveurs autorisés
```

```
for host_i in $SMBHOSTOUT; do
 $IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -d $host_i --dports
netbios-dgm -j ACCEPT
done

# ICMP
# Accepter toutes les connexions TCP entre hôtes du trésor
$IPTABLES -A FORWARD -p icmp -i $OURDEV -s $OURNET -d $OURNET -j ACCEPT

# ICMP - INCOMING
# We will allow ICMP datagrams in of the allowed types.
for type_i in $ICMPIN; do
 $IPTABLES -A FORWARD -p icmp -i $ANYDEV -d $OURNET --icmp-type $type_i
-j ACCEPT
done

# ICMP - OUTGOING
# We will allow ICMP datagrams out of the allowed types.
for type_i in $ICMPOUT; do
 $IPTABLES -A FORWARD -p icmp -i $OURDEV -d $ANYADDR --icmp-type $type_i
-j ACCEPT
done

if [ "$LOGGING" ]
then
        # Log barred TCP
        $IPTABLES -A FORWARD -m tcp -p tcp -j $DROPMODE
        # Log barred UDP
        $IPTABLES -A FORWARD -m udp -p udp -j $DROPMODE
        # Log barred ICMP
        $IPTABLES -A FORWARD -m icmp -p icmp -j $DROPMODE
fi

# Redirige et masque les connexion vers smtp et pop3 sur le serveur mail
SNDI
$IPTABLES -t nat -A PREROUTING -i $OURDEV -p tcp -m multiport -d
10.6.9.88 --destination-port pop3,smtp -j DROP
$IPTABLES -t nat -A PREROUTING -i $OURDEV -p tcp -s $OURNET --dport pop3
-j DNAT --to 10.6.9.88:110
$IPTABLES -t nat -A PREROUTING -i $OURDEV -p tcp -s $OURNET --dport smtp
-j DNAT --to 10.6.9.88:25
$IPTABLES -t nat -A POSTROUTING -o $ANYDEV -p tcp -m multiport -s
$OURNET -d 10.6.9.88 --destination-port pop3,smtp -j MASQUERADE

# nat pour surfer sur plusieurs machines avec le même compte proxy
$IPTABLES -t nat -A POSTROUTING -p tcp -s neo -d proxy.sndi-ci.com
--dport squid -j MASQUERADE

#
# sauve la configuration
iptables-save >/etc/sysconfig/iptables
```

8.1. Shorewall

Shorewall est un parefeu/routeur/nat basé sur iptables, et implémenté sous la forme d'un ensemble de scripts bash. Il permet de configurer relativement simplement un parefeu. Les fichiers de configurations se trouvent dans `/etc/shorewall`

- le fichier `zones` décrit les zones du parfeu:
  - `local`: le réseau local (la zone interne)
  - `net`: le réseau internet (la zone externe et hostile)
  - `dmz`: la zone démilirarisée (la zone interne on protégée par parefeur)
- le fichier `interfaces` associe les interfaces aux zones

```
loc     eth0    192.168.0.255
net     ppp0    -
```

- le fichier `policy` donne les grandes règles générale (tout protos confondus) de gestion

```
loc             all             ACCEPT
fw              all             ACCEPT
net             all             DROP            info
all             all             REJECT          info
```

- le fichier `rules` donne les règles de filtrages au cas par cas

```
ACCEPT  net     fw      tcp
1214,2234,4444,4448,4661:4662,6346,6699,6882,9999       -
ACCEPT  net     fw      udp     4448,4665,4666  -
```

- le fichier `masq` donne les règles pour le masquerading (partage d'une seule adresse ip à plusieurs)

```
ppp0    192.168.0.0/24
```