Architecture Sécurisée

Sommaire

Ce stage a pour but de fournir les bases techniques, à la fois théoriques et pratiques, qui sont nécessaires à la mise en place d'une passerelle Internet sécurisée.
À ce titre, il constitue une introduction aux techniques et problématiques que peuvent rencontrer les administrateurs réseau dans leur tâche de sécurisation d'un système d'information plutôt qu'une référence figée sur l'état de l'art en la matière.

Ce stage est organisé sous forme de cours et de TP qui permettront d'expliciter les notions fondamentales et le fonctionnement des outils utilisés et de mettre immédiatement ces connaissances en pratique.

L'ordre des sections qui suivent respecte, à quelques différences près, celui du déroulement du stage.

9. **Conclusion**
   Quelques considérations en guise de conclusion.

Conventions

Dans les pages, la convention suivante est utilisée :

- Les commandes sont notées en **`rouge`** ;
- Les répertoires ou les noms de fichiers sont notés en **`vert`**. Les noms répertoires se terminent par / ;
- Les paramètres de fichiers de configuration sont notés en **`violet`**.

Présentation de l'architecture

Le stage peut accueillir 16 personnes, réparties en 8 équipes de deux personnes.
T désigne le numéro de l'équipe, de 1 à 8 pour les stagiaires et le n°9 pour l'intervenant.

Matériel disponible par équipe :

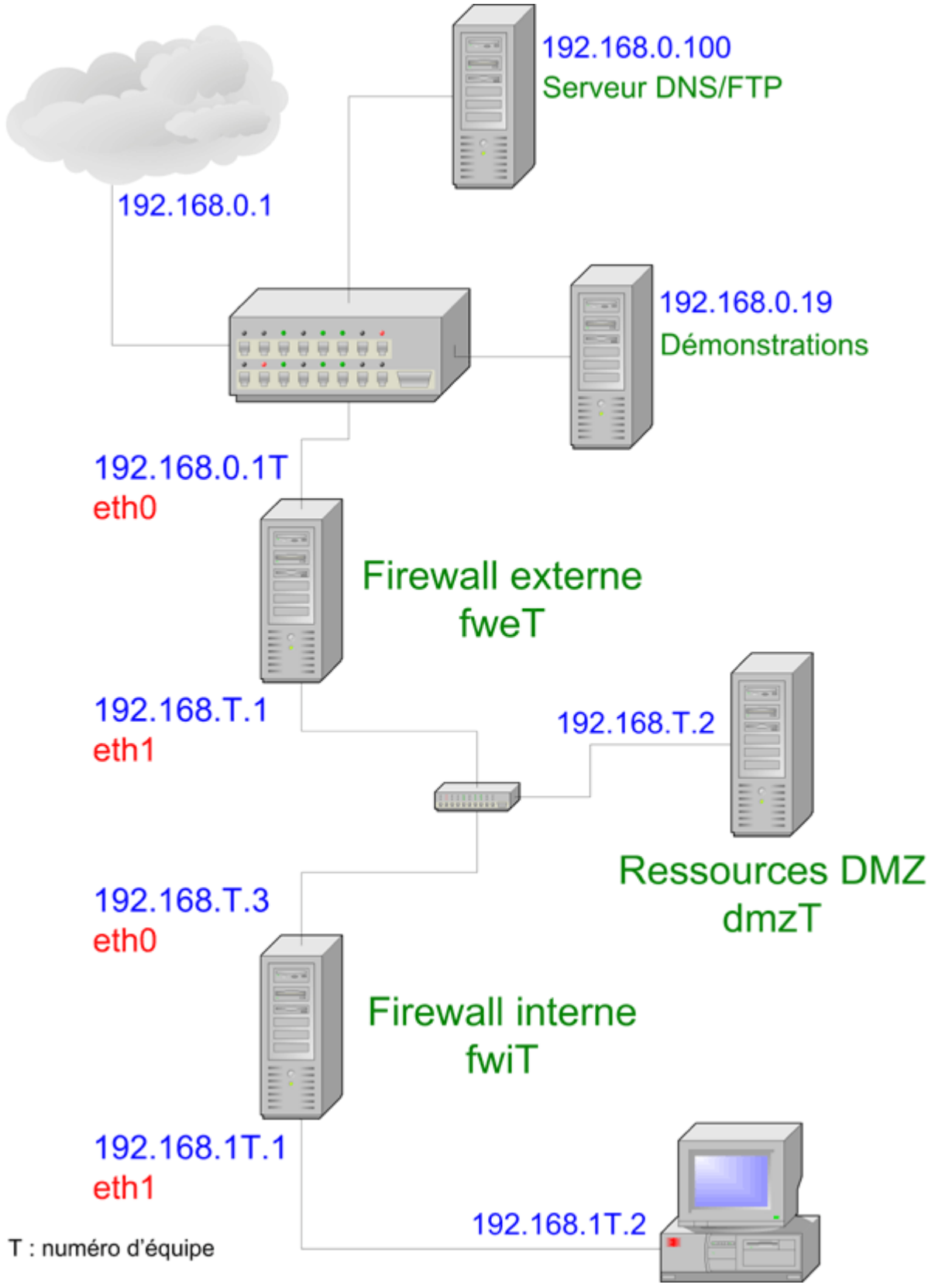
- 4 PCs
- 1 Hub, appelé hub DMZ
- 1 switch écran/clavier + 1 console (clavier, écran, souris)

Matériel commun :

- Un hub 24 ports, appelé hub central
- **Un serveur FTP**
- Une machine de démonstration sur laquelle l'intervenant réalise les opérations présentées

Le réseau commun (192.168.0.0/24) figure de manière abstraite le réseau public. Ainsi, une communication entre les firewalls externes de deux groupes sera assimilée à une communication sur Internet. Par la suite, le réseau 192.168.0.0/24 sera désigné comme le "réseau public" pour simplifier les explications.

Schéma du réseau



Détails des machines

- firewall externe (FWE): cette machine est positionnée en frontal sur Internet. Son rôle est de filtrer les connections entrantes et de les rediriger sur les machines de la DMZ. Il s'agit d'un **PC sous Linux (noyau 2.6)**, avec deux cartes réseau. Il est connecté au réseau public et au hub DMZ par deux câbles droits.
- machine démilitarisée (DMZ): Il s'agit d'un **PC sous OpenBSD 3.7** destiné à accueillir les différents services offerts aux utilisateurs distants. Il dispose d'une carte réseau, connectée au hub DMZ par un câble droit.
- firewall interne (FWI): Il s'agit d'un **PC en double boot sous Linux (noyau 2.6) et Windows Server 2003** avec deux cartes réseau. Ce double boot Linux / Windows servira à montrer l'interopérabilité entre les deux implémentations IPsec. Concernant Linux, le choix de la famille des noyaux 2.6.x est motivé par sa gestion native d'Ipsec. Ce PC est connecté au hub dmz par un câble droit et à la machine interne par un câble croisé.
- machine du réseau interne (LAN): Il s'agit d'un **PC sous Windows XP** avec une carte réseau, qui joue le rôle de poste client dédié à la bureautique. Il est connecté au firewall interne par un câble croisé.

Le choix de systèmes d'exploitation variés vise à illustrer la diversité de l'offre tout en identifiant par la pratique les particularités de chaque famille de systèmes.

Installation des firewalls (FWE et FWI) sous Linux

Cette partie décrit l'installation des firewalls FWI et FWE. Les systèmes ont été choisis quasi identiques par souci de simplicité dans le cadre du TP. En production, une bonne défense en profondeur se traduit plutôt par l'installation de deux systèmes différents, de façon à ce que la découverte d'une vulnérabilité sur l'un d'eux n'affecte pas directement l'ensemble du dispositif de filtrage.

Il est important de construire le pare-feu sur la base d'un système minimaliste dans la mesure où chaque service supplémentaire est un facteur d'exposition aux attaques.

La distribution de base utilisée dans le cadre du TP est la Mandrake 10.

Les outils requis dans le cadre du TP sont ceux de configuration et d'analyse réseau (iptables, tcpdump, ...) et ceux de développement (pour la compilation du noyau). L'installation du serveur graphique est à éviter mais admise dans le cadre du TP. Dans une configuration opérationnelle, on pourra également supprimer les outils de développement (compilation sur une plate-forme séparée) et d'analyse réseau.

Note

Dans quel but recommander une installation minimaliste, sans mode graphique ni outils superflus ? Il s'agit d'une mesure permettant d'assurer une défense en profondeur.

Le premier niveau de sécurité est constitué par la minimisation du nombre de services activés, ainsi que par la configuration du système et des services autorisés. Ce premier niveau vise à faire respecter la politique de sécurité et à assurer qu'un matériel (ici, le firewall) fonctionnera

conformément au besoin exprimé.
Le second niveau de sécurité vise à prendre un ensemble de précautions pour restreindre les possibilités d'un attaquant qui aurait, par exploitation d'une vulnérabilité, réussi à compromettre un des équipements du réseau. En règle générale, il s'agit de voir que chaque outil superflu ne peut servir qu'à un utilisateur non autorisé: élévation locale de privilèges, lancement d'attaques à destination d'autres équipements, etc.

Typiquement, dans le cas d'un serveur web, le premier niveau de sécurité est atteint par une configuration adaptée du service et, le cas échéant, par l'application de règles de filtrage locales. La prise en compte du second niveau consiste à supprimer tous les outils ne contribuant pas au service fourni, en commençant par les plus intéressants pour un attaquant introduit sur le serveur (compilateur, interpréteur de langage de haut niveau, service réseau non utilisé, etc.), puis en étendant cette démarche à l'ensemble des exécutables non indispensables.

Cette recherche de sécurité en profondeur (minimisation du niveau d'exposition *et* des risques en cas de compromission), à laquelle peuvent s'ajouter des considérations de performances, motivent l'installation d'un système minimaliste et la recompilation du noyau.

Déroulement de l'installation

L'installation du système Linux s'effectue à partir d'une disquette de boot et du serveur FTP du TP. Les écrans d'installation sont en principe assez explicites. Les paragraphes suivants reprennent la liste des choix à effectuer dans le cadre du TP.

Configuration pré-install

NB : les paramètres réseau (IP, ...) fournis ici correspondent au FWE. Les indications fournies sur le schéma d'architecture réseau permettent leur adaptation au FWI.

La première interface (lancée depuis la disquette) est de type texte/ncurses.

- source d'installation : **ftp**
- interface réseau à configurer : **eth0** *(insérer alors la seconde disquette, qui contient les pilotes)*
- adresse IP : **192.168.0.1T** *(où T est le numéro de groupe)* les autres paramètres réseau (masque, broadcast, ...) s'en déduisent facilement
- nom d'hôte : **fweT**
- proxy ftp : **aucun**
- utilisateur ftp : *vide* (service FTP anonyme)
- mot de passe : *vide*

Les outils d'installation sont alors téléchargés depuis le serveur FTP.

La seconde interface est de type graphique/X11.

- choix de la langue : **français**
- souris : **PS2 à roulette** *(puis test souris)*
- clavier : **français**
- niveau de sécurité : **standard** *(pas de mail ou root@localhost, il s'agit de l'adresse à laquelle sont envoyés des alertes et certains rapports de logs)*

- partitionnement personnalisé (à propos du partitionnement, voir les remarques complémentaires à la fin de cette section)
  effacer toutes les partitions existantes
  créer une partition de type Ext3 de 5 Go montée sur '/'
  créer une partition de type swap de 512 Mo
- choix des paquetages *(pour simplifier la tâche: pas de sélection individuelle)*
  Clients réseau
  Utilitaires console
  Développement
  Documentation (facultatif)
  Pare-feu/routeur
  Station KDE
  (les paquetages sont alors téléchargés et installés)
- mot de passe root : **xxxxxxxx**
- compte utilisateur (login/mot de passe) : **cfssi/xxxxxxxx**
- désactiver l'ouverture automatique de session
- amorçage sur le MBR

On arrive alors sur un écran résumant la configuration. Quelques réglages doivent encore être effectués.

- réseau :
  Autodétection
  Connection LAN
  eth0: déjà bien configurée
  eth1: IP=192.168.T.1, Netmask=255.255.255.0
  config hôte: pas de modification
- interface graphique :
  Ecran plug-and-play bien détecté
  Carte graphique bien détectée
  Configurer seulement la carte Matrox
  XFree 4.3 avec (ou sans) accélération 3D
  RAM bien détectée
  Résolution: bonne par défaut (adaptable pour le confort visuel)
  Désactiver le démarrage en mode graphique
- services à activer par défaut : **crond, keytable, iptables, local, network, numlock, random, syslog**
- pas de téléchargement de mises à jour

L'installation se termine et on redémarre la machine.

Configuration post-install

Il reste quelques réglages supplémentaires à effectuer sur la machine.

Au démarrage, choisir le système Linux, puis se connecter en tant que root.
Attention: En mode graphique, ne *jamais* démarrer le gestionnaire de fenêtres en tant que root (principe de minimisation des privilèges, donc des risques, compte tenu de la complexité de ce service). Il faut donc se connecter via un compte non privilégié puis lancer un terminal graphique (type Xterm) et y exécuter **su** pour obtenir des privilèges à l'intérieur d'un terminal texte.
Au vu de la faiblesse du cloisonnement entre applications qu'offre le serveur graphique, ce compte doit être convenablement protégé. Aucune application susceptible d'être compromise ne doit être

exécutée, en particulier lorsqu'un terminal root est lancé. L'administration en mode texte permet dans une large mesure de contourner ces difficultés.

Il est recommandé de désactiver la fonction supermount. Il s'agit d'un mode de gestion plus convivial pour l'accès aux périphériques de stockage amovibles, qui est géré par un patch noyau spécifique à Mandrake. Son utilisation pose donc des problèmes en cas de recompilation d'un noyau à partir des sources standard non patchées Mandrake.

Pour désactiver le supermonteur au démarrage :

```
$> supermount -i disable
```

On peut alors procéder aux ajustements du fichier **/etc/fstab** des points de montage du système de fichiers. Le fichier doit doit ressembler à ceci :

```
/dev/hda1       /               ext3    defaults                        0
0

/dev/hda5       swap            swap    defaults                        0
0

/dev/hdc        /mnt/cdrom      auto    ro,user,nodev,nosuid            0
0

/dev/fd0        /mnt/floppy     auto    ro,user,nodev,nosuid            0
0

none            /proc           proc    defaults                        0
0
```

Attention : lorsque certains périphériques (disquette, CDROM) peuvent être montés par des utilisateurs non privilégiés (option "user"), il est important d'exclure l'importation de fichiers spéciaux susceptibles de permettre une élévation locale de privilèges (options "nodev" et "nosuid").

On peut alors remonter les partitions avec les nouvelles options:

```
$> umount -a

$> mount -a
```

et observer l'état courant des montages par **mount** et **df** .

La suite passe en revue les fichiers de configuration réseau (résolution de noms puis configuration des interfaces).

/etc/nsswitch.conf

```
passwd/shadow/group : ne garder que "files"

hosts : ne garder que "files" et "dns" (dans cet ordre)

bootparams : infos pour clients sans disque,supprimer

netgroup+publickey : pour NIS-NFS sécurisés, supprimer
```

```
ethers/netmasks/networks/protocols/rpc/services : laisser à "files"

automount/aliases : ne garder que "files"
```

NB : On peut supprimer "dns" sur une passerelle opérationnelle, un filtre IP n'est pas censé générer de trafic.

/etc/resolv.conf

```
nameserver 192.168.0.100
```

/etc/host.conf

```
order hosts,bind

multi on
```

/etc/hosts

```
192.168.0.1T    fweT

192.168.T.1     fweT

127.0.0.1       localhost.localdomain localhost
```

/etc/sysconfig/network

```
HOSTNAME=fweT

NETWORKING=yes

GATEWAY=192.168.0.1
```

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0

BOOTPROTO=static

IPADDR=192.168.0.1T

NETMASK=255.255.255.0

NETWORK=192.168.0.0

BROADCAST=192.168.0.255

ONBOOT=yes

MII_NOT_SUPPORTED=yes
```

Cette dernière ligne permet de configurer l'interface même si le câble réseau est vu comme débranché.

/etc/sysconfig/network-scripts/ifcfg-eth1

```
DEVICE=eth1
```

```
BOOTPROTO=static

IPADDR=192.168.T.1

NETMASK=255.255.255.0

NETWORK=192.168.T.0

BROADCAST=192.168.T.255

ONBOOT=yes

MII_NOT_SUPPORTED=yes
```

Quelques commandes utiles :

- **`/etc/init.d/network restart`** : cette commande désactive puis réactive les interfaces et la configuration réseau. Elle est utile pour prendre en compte un nouveau paramètre réseau sans redémarrer la machine.
- **`route -n`** ou **`netstat -rn`** : ces deux commandes sont équivalentes (la seconde est toutefois plus "portable") et affichent les tables de routage actives.
- **`ifconfig -a`** : affiche la configuration de toutes les interfaces réseau (actives ou non).

Il faut également ajuster quelques paramètres noyau au démarrage via le fichier **`/etc/sysctl.conf`**. Les fonctions de routage et de détection de spoofing d'adresse source s'activent par :

```
net.ipv4.ip_forward = 1

net.ipv4.conf.default.rp_filter = 1

net.ipv4.conf.all.rp_filter = 1

le reste est inchangé
```

L'activation à chaud du routage s'effectue par la commande **`sysctl -w net.ipv4.ip_forward=1`**.

Enfin, on pourra utiliser la commande Mandrake **`urpmi`** pour récupérer quelques utilitaires non installés par défaut :

```
$> urpmi bind-utils

$> urpmi tcpdump

$> urpmi ethereal
```

Remarques complémentaires

Cette installation se veut la plus simpliste possible. Pour un firewall de production, certains choix doivent être adaptés.

En ce qui concerne le partitionnement du disque, il est en général conseillé de monter des partitions séparées sur les répertoires susceptibles de grossir de façon mal contrôlée (/var à cause des logs, /home et /tmp à cause des utilisateurs, ...). Cela évite le blocage du système en cas de saturation de l'une de ces partitions. Par ailleurs, la séparation de /home et /root facilite la

récupération des comptes lors des réinstallations. La séparation de /boot permet de contourner les limitations de certains OS sur l'offset du noyau dans le disque ou encore d'installer plusieurs distributions Linux sur un même disque (mettre le /boot en commun). Enfin, la séparation de /usr et /usr/local peut permettre de partager des binaires entre machines distinctes et/ou d'envisager des montages en lecture seule.

Le principe de minimisation conduit également à préférer une sélection individuelle des paquetages à installer. On évitera, par exemple, d'installer un compilateur (gcc, g++, ..) ou un interpréteur avancé (perl, python, ...) sur un firewall relié à Internet. De même, l'interface graphique n'a pas sa place sur un serveur ou un routeur.

Recompilation du noyau

Motivation

Comme indiqué précédemment, le système se doit d'être minimaliste pour éviter les expositions inutiles et pour réduire les possibilités d'un attaquant qui arriverait malgré tout à le compromettre.

Il est donc logique de recompiler le noyau pour adapter les fonctionnalités embarquées et supprimer celles qui ne sont pas nécessaires (ou, dans certains cas, en ajouter). Une autre conséquence positive est que la recompilation modifie la structure interne du binaire, ce qui, dans certains cas, peut faire échouer les attaques standard car elles ne sont plus adaptées.
NB: Le noyau étant, entre autres choses, responsable de la gestion des privilèges et des périphériques, toute vulnérabilité est de nature à compromettre directement l'ensemble du système.

Dans la plupart des distributions, beaucoup de fonctionnalités sont activées par défaut dans le noyau.
Ceci vise à une meilleure compatibilité avec un besoin opérationnel et une plate-forme matérielle particulière inconnue à priori par l'éditeur, qui cherche donc la couverture la plus large possible. La plupart de ces fonctionnalités sont inutiles dans le cas d'un firewall. Il convient donc de les supprimer.

En règle générale, une fonctionnalité peut être intégrée au noyau Linux de deux façons : ou elle fait partie intégrante du noyau, ce qui augmente sa taille, ou elle est compilée séparément sous forme d'un module chargeable (et déchargeable) à la demande.

Pour limiter les possibilités de compromission du noyau en cours de fonctionnement, la configuration retenue pour le TP est minimaliste et monolithique : on ne conserve que les fonctionnalités strictement nécessaires et on les inclut directement dans le noyau. De plus, le support des modules est désactivé pour empêcher l'introduction de modules indésirables dans l'espace noyau.

Le noyau Linux utilisé dans le cadre du TP est de la famille 2.6. A ce titre, il gère nativement les protocoles et les structures de données utilisés par IPsec et intègre certaines primitives cryptographiques.

Réalisation pratique

Récupérer l'archive du noyau 2.6.4 sur le serveur FTP et la copier dans **`/usr/src/`** :

```
$> cd /usr/src

$> ftp 192.168.0.100

$> get pub/kernel/linux-2.6.4.tar.gz

$> quit

$> tar xzf linux-2.6.4.tar.gz

$> ln -s linux-2.6.4 linux

$> cd linux

$> make menuconfig
```

Il faut notamment supprimer le support des modules et sélectionner les pilotes adaptés au matériel ainsi que les fonctions liées à Netfilter et à IPsec.
En cas de problème, voici un exemple de fichier **.config** utilisable pour un pare-feu dans le cadre du TP. Il suffit de le copier dans **/usr/src/linux/** et d'exécuter la commande **make oldconfig** pour prendre en compte ses paramètres.

La compilation du noyau s'effectue alors par **make**, qui équivaut à **make dep && make bzImage** (**&& make modules** si le support des modules est activé). Il ne reste alors qu'à installer le noyau, ce qui peut s'effectuer automatiquement par **make install** ou manuellement par :

```
$> cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.4

$> cp .config /boot/config-2.6.4

$> cp System.map /boot/System.map-2.6.4
```

Seul le noyau lui-même (bzImage) est indispensable. Les deux autres fichiers fournissent des informations simplifiant la maintenance ou le traçage du système (un attaquant peut reconstituer ces informations à travers **/proc/**).

NB : Le noyau pourra monter la partition racine sans réaliser d'opération préalable particulière (chargement de module, etc.), il n'est donc pas utile de générer un *initrd*.

Dans le cas d'un noyau modulaire, il aurait aussi fallu exécuter:

```
$> make modules_install
```

Il faut assurer la bonne prise en compte du nouveau noyau par le chargeur de boot LILO. Le fichier **/etc/lilo.conf** doit ressembler à :

```
boot=/dev/hda

map=/boot/map

vga=normal

default="linux-2.6"

keytable=/boot/fr-latin1.klt
```

```
prompt

nowarn

timeout=50

message=/boot/message

menu-scheme=wb:bw:wb:bw

image=/boot/vmlinuz-2.6.3-4mdk

        label="linux"

        root=/dev/hda1

        initrd=/boot/initrd-2.6.3-4mdk.img

        append="devfs=mount acpi=ht resume=/dev/hda6 splash=silent"

        read-only

image=/boot/vmlinuz-2.6.4

        label="linux-2.6"

        root=/dev/hda1

        read-only

        append="nomodules"
```

La commande **lilo** permet d'appliquer cette configuration au chargeur de boot.

Il faut enfin vider les fichiers **/etc/modules** et **/etc/modules.conf** .

Il suffit alors de redémarrer la machine en choisissant le noyau linux-2.6 pour lancer le nouveau noyau.

Suppression des services inutiles

Introduction

Sous Linux, tous les processus (à l'exception de ceux lancés par le noyau pour prendre en compte un périphérique branché à chaud ou charger un module dont il a besoin) descendent d'un processus initial, "init", qui est lancé au démarrage par le noyau une fois la partition racine montée. Init sélectionne alors les services à démarrer en fonction d'un paramètre appellé niveau d'exécution (runlevel).

Il existe 7 runlevels différents numérotés de 0 à 6 :
0 - halt : le système passe dans ce niveau lorqu'il reçoit l'ordre d'éteindre la machine
1 - Single user mode : le système ne gère qu'un seul utilisateur (root)
2 - Multiuser, without NFS : système multi-utilisateur mais où les systèmes de fichiers réseau ne

sont pas montés
3 - Full multiuser mode : système multi-utilisateur complet en mode console
4 - unused : pas de sens prédéfini
5 - X11 : système multi-utilisateur complet avec démarrage automatique du service graphique X11
6 - reboot : le système passe dans ce niveau lorqu'il reçoit l'ordre de redémarrer la machine

Le runlevel par défaut est défini dans le fichier **/etc/inittab** par une ligne de la forme : id:3:initdefault: (ici, le runlevel 3). Ce fichier permet aussi de paramétrer le nombre de consoles virtuelles de login en mode texte, le comportement du système sur certains évènements (réception d'un Ctrl-Alt-Del, d'un message d'un onduleur, etc.).

La commande **runlevel** permet de connaître le runlevel courant. À tout moment, l'utilisateur root peut changer de runlevel en tapant la commande **init n**, où n est le numéro du runlevel qu'il souhaite atteindre. Les runlevels 0 et 6 sont généralement atteints via les commandes **halt**, **reboot** ou **shutdown**, ou encore par Ctrl-Alt-Del.

Les services

Lors du passage d'un runlevel à un autre, le système lance ou arrête un certain nombre de services.

Voici une liste des services typiquement disponibles sur une machine Linux :

alsa	internet	nfslock	rawdevices
apmd	iptables	numlock	saslauthd
atd	keytable	partmon	sound
crond	kheader	portmap	syslogd
cups	killall	postfix	usb
dm	netfs	proftpd	xfs
harddrake	network	random	xinetd

Chacun de ces services est activable par un script qui se trouve dans le répertoire **/etc/rc.d/init.d/** .

On peut lancer ou arrêter un service au moyen de l'API standard des services.

On utilise ces scripts avec une instruction du type :

```
$> /etc/init.d/nom_du_script commande
```

où les commandes les plus courantes sont : **start stop restart status**

La liste des services à lancer pour chaque runlevel se trouve dans les répertoires **/etc/rc0.d/** , **/etc/rc1.d/** , **/etc/rc2.d/** , **/etc/rc3.d/** , **/etc/rc4.d/** , **etc/rc5.d/** , **/etc/rc6.d** , le numéro correspondant au runlevel.

Voici en détail l'un d'entre eux :

```
ls /etc/rc3.d

K09dm           S14nfslock      S40saslauthd       S80postfix

S03iptables     S17alsa         S56xinetd          S85numlock
```

```
S05harddrake    S18sound        S60cups         S85proftpd

S10network      S20random       S75keytable     S95kheader

S11portmap      S25netfs        S76internet     S99local

S12syslogd      S29apmd         S78crond

S13partmon      S40atd
```

Tous ces fichiers sont des liens symboliques pointant sur les scripts vus précédemment (ou sur /etc/rc.local pour le dernier).

Le système les interprète comme suit :

- Un lien nommé KNNservice indique qu'il faut arrêter le service en question lorsque l'on rentre dans ce runlevel (K=kill), sauf s'il n'était pas actif.
- Un lien nommé SNNservice indique qu'il faut démarrer le service en question lorsque l'on rentre dans ce runlevel (S=start), sauf s'il était déjà actif.

Les numéros servent à indiquer au système dans quel ordre il devra effectuer ces actions : les arrêts se font toujours avant les lancements, et chaque série suit la numérotation des liens (et l'ordre lexicographique en cas d'égalité).

Configuration pour le TP

Les services à démarrer sont les suivants :

- iptables: chargement des règles de filtrage IP
- network : activation des interfaces réseau
- syslogd : activation de la journalisation système (/var/log/messages)
- random : chargement de la graine du générateur d'aléa (chaînage entre l'arrêt et le redémarrage de la machine)
- keytable : prise en charge du clavier francais
- numlock: verrouillage du pavé numérique
- crond : lancement du planificateur de tâches (facultatif)
- local : prise en compte des commandes particulières (/etc/rc.local)

Attention, il est important de charger les règles de filtrage *avant* d'activer les interfaces réseau pour éviter les états transitoires vulnérables.

Remarque : il existe aussi des utilitaires permettant de configurer ces services en mode graphique: ksysv (KDE), drakxservices (Mandrake), etc. Il s'agit en fait de simples frontaux graphiques qui manipulent les liens symboliques décrits plus haut.

Vérification de la configuration

La commande **`ps auxw`** permet de lister les processus lancés (les processus dont le nom figure entre crochets [] correspondent en principe aux threads noyau). Il est important de minimiser cette liste, de configurer et de connaître l'utilité de chaque processus effectivement actif.

La commande **netstat -anp --inet** permet de lister les sockets réseau ouvertes par les différents processus. Là encore, il est primordial d'éliminer tous les serveurs non désirés et de sécuriser les autres.

<<< Précédent
Installation des firewalls (FWE et FWI)

Suivant >>>
Filtrage IP sur FWE

Sommaire

Installation d'OpenBSD sur la DMZ

Le choix a été fait d'installer le système OpenBSD sur la machine DMZ. Il s'agit d'un système libre de type Unix. Pour être plus précis, il fait partie de la famille des BSD, inspirée du système Unix développé par l'université de Berkeley.

Il a en outre une caractéristique fort importante : il est axé sécurité. Par exemple, le serveur Apache inclus dans le système est chrooté et tourne sous un utilisateur non privilégié.

Pour ces raisons, ce système semble particulièrement adapté à l'exposition de services sur Internet.

Installation d'OpenBSD 3.7

Le système est installé à partir du serveur ftp.
Booter à partir de la disquette. L'interface d'installation est en mode texte et suffisante.
Voici une description des éléments clés de l'installation.

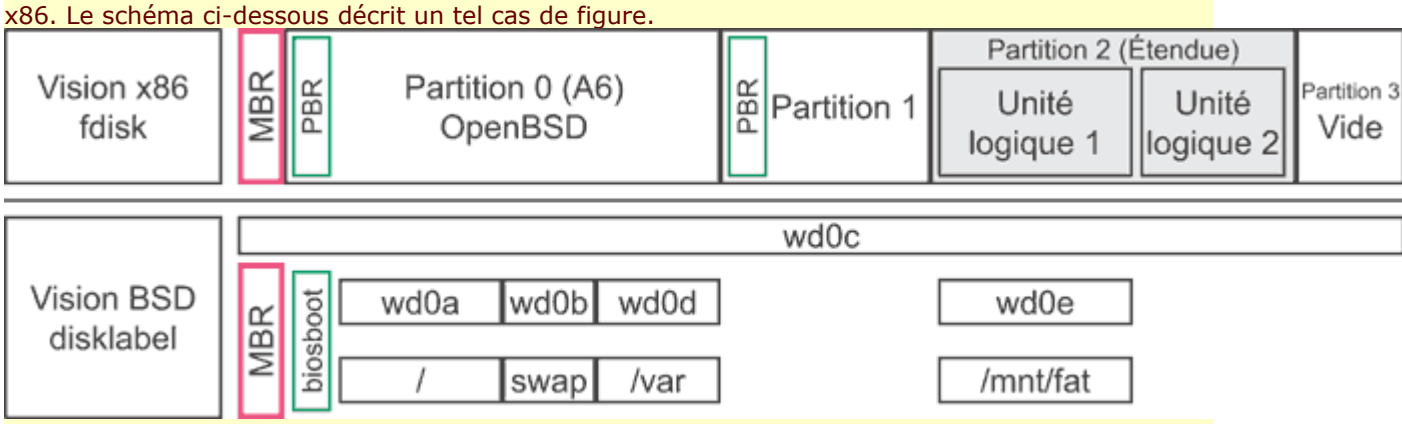
```
(I)nstall, (U)pgrade or (S)hell? i


Welcome to the OpenBSD/i386 3.7 install program.


Terminal type: [vt220] <Entrée>

Do you wish to select a keyboard encoding table? [no] y

Select your keyboard type: (P)C-AT/XT, (U)SB or 'done' [P] <Entrée>

The available keyboard encoding tables are:


        be br de dk es fr it jp lt no pt ru sf sg sv ua uk us


Table name? (or 'done') [us] fr
```

```
kbd: keyboard mapping set to fr



Proceed with install? [no] y
```

La plupart du temps, les valeurs proposées par défaut sont les bonnes.
C'est le cas avec le type de terminal, en revanche, on peut choisir le français comme table de codage de caractères.

Arrive le partitionnement des disques. Il faut entendre le terme partition à deux "niveaux".
Les partitions au sens classique, c'est-à-dire le partage du disque selon la norme x86 en partitions de différentes type (FAT32, NTFS, OpenBSD, Ext3 ou étendue par exemple). On utilise pour cela l'outil fdisk.

Et les partitions au sens OpenBSD : les labels. C'est sous cette forme que OpenBSD voit le découpage du disque. Dans le cas d'une machine x86, on utilise deux types de partitionnement en parallèle. Sous fdsik, on désigne la partition OpenBSD avec l'identifiant A6 pour éviter que ces autres systèmes n'écrivent dessus. Il faut bien sûr faire correspondre les labels au partitionnement x86. Le schéma ci-dessous décrit un tel cas de figure.



Pour manipuler les labels BSD, on utilise l'outil disklabel. Certains labels sont réservés. Le label **c** désigne la totalité du disque, le label **a** désigne la partition racine et le label **b** la première partition de swap. Les labels sont préfixés par le type de support (wd pour un disque IDE, sd pour un disque SCSI) et par le numéro du périphérique sur son bus. Dans le cas d'un disque IDE seul, la partition racine sera donc désignée par le label wd0a.

```
Cool!  Let's get to it...

You will now initialize the disk(s) that OpenBSD will use. To enable all

available security features you should configure the disk(s) to allow the

creation of separate filesystems for /, /tmp, /var, /usr, and /home.



Available disks are: wd0.

Which one is the root disk? (or done) [wd0] <Entrée>
```

On utilise le disque par défaut.

```
Do you want to use *all* of wd0 for OpenBSD? [no] <Entrée>
```

Dans un premier temps, on crée une partition BSD en utilisant l'outil fdisk classique.

```
You will now create a single MBR partition to contain your OpenBSD data.
This

partition must have an id of 'A6'; must *NOT* overlap other partitions; and

must be marked as the only active partition.



he 'manual' command describes all the fdisk commands in detail.



Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]

Offset: 0       Signature: 0xAA55
           Starting       Ending       LBA Info:

 #: id    C   H  S -    C   H  S [       start:      size   ]

------------------------------------------------------------------------

*0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB

 1: 00    0   0  0 -    0   0  0 [           0:           0 ] unused

 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused

 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused

Enter 'help' for information



fdisk: 1> help (pour afficher une aide succincte)



fdisk: 1> e 0

            Starting       Ending       LBA Info:

 #: id    C   H  S -    C   H  S [       start:      size   ]

------------------------------------------------------------------------

 1: 00    0   0  0 -    0   0  0 [           0:           0 ] unused

Partition id ('0' to disable)  [0 - FF]: [0] (? for help) 0 (on efface la
partition existante)



fdisk: 1> e 0
```

```
Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 1: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
Partition id ('0' to disable)  [0 - FF]: [0] (? for help) a6
Do you wish to edit in CHS mode? [n] n
BIOS Starting cylinder [0 - 2585]: [0] 63
Label size [0 - 239]: [0] 5G

fdisk:*1> f 0
Partition 0 marked active.

fdisk:*1> p
Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S  [       start:      size   ]
------------------------------------------------------------------------
*0: A6    0   1  1 -  2585 239 63 [          63:  5056039600 ] OpenBSD
 1: 00    0   0  0 -    0   0  0  [           0:           0 ] unused
 2: 00    0   0  0 -    0   0  0  [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0  [           0:           0 ] unused

fdisk:*1> w
Writing MBR at offset 0.
wd0: no disk label

fdisk: 1> q
```

Deux choses importantes : l'identifiant d'une partition BSD est A6 et la première partition commence au secteur n°63, les secteurs précédents étant réservés.

Vient ensuite l'éditeur disklabel, dans lequel il faut définir les partitions OpenBSD.

```
You will now create an OpenBSD disklabel inside the OpenBSD MBR
partition. The disklabel defines how OpenBSD splits up the MBR partition
into OpenBSD partitions in which filesystems and swap space are created.

The offsets used in the disklabel are ABSOLUTE, i.e. relative to the
start of the disk, NOT the start of the OpenBSD MBR partition.
disklabel: no disk label
WARNING: Disk wd0 has no label. You will be creating a new one.

# using MBR partition 1: type A6 off 3069360 (0x2ed5b0) size 36030960
(0x225c9f0)

Treating sectors 3069360-39100320 as the OpenBSD portion of the disk.
You can use the 'b' command to change this.
Initial label editor (enter '?' for help at any prompt)
# ?
Available commands:
      p [unit]  - print label.
      M         - show entire OpenBSD man page for disklabel.
      e         - edit drive parameters.
      a [part]  - add new partition.
      b         - set OpenBSD disk boundaries.
      c [part]  - change partition size.
      d [part]  - delete partition.
      D         - set label to default.
      g [d|b]   - Use [d]isk or [b]ios geometry.
```

```
      m [part]  - modify existing partition.

      n [part]  - set the mount point for a partition.

      r         - recalculate free space.

      u         - undo last change.

      s [path]  - save label to file.

      w         - write label to disk.

      q         - quit and save changes.

      x         - exit without saving changes.

      X         - toggle expert mode.

      z         - zero out partition table.

      ? [cmnd]  - this message or command specific help.

Numeric parameters may use suffixes to indicate units:

      'b' for bytes, 'c' for cylinders, 'k' for kilobytes, 'm' for
megabytes,

      'g' for gigabytes or no suffix for sectors (usually 512 bytes).

      Non-sector units will be rounded to the nearest cylinder.

Entering '?' at most prompts will give you (simple) context sensitive help.
```

Dans le cadre de ce TP, il faut faire au plus simple : une partition principale et un partition swap.

```
# d a


# a a

offset: [3069360] <Entrée>

size: [36030960] 4G

Rounding to nearest cylinder: 307440

FS type: [4.2BSD] <Entrée>

mount point: [none] /


# a b
```

```
offset: [3376800] <Entrée>

size: [35723520] 512M

Rounding to nearest cylinder: 614880

FS type: [swap] <Entrée>
```

En sortant de cet éditeur, une étape de confirmation des points de montage est là pour éviter toute erreur.

Il faut ensuite choisir le nom d'hôte de la machine.

```
System hostname? (short form, e.g. 'foo') dmzT
```

Puis viens la configuration du réseau.

```
Configure the network? [yes] <Entrée>

Available interfaces are: fxp0.

Which one do you wish to initialize? (or 'done') [fxp0] <Entrée>

Symbolic (host) name for de0? [dmzT] <Entrée>

The media options for de0 are currently

        media: Ethernet autoselect (100baseTX)

Do you want to change the media options? [no] <Entrée>

IPv4 address for de0? (or 'none' or 'dhcp') 192.168.T.2

Netmask? [255.255.255.0]

No more interfaces to initialize.

DNS domain name? (e.g. 'bar.com') [my.domain] groupeT.tp

DNS nameserver? (IP address or 'none') [none] 192.168.0.100

Use the nameserver now? [yes] n

Default IPv4 route? (IPv4 address, 'dhcp' or 'none') 192.168.T.1

add net default: gateway 192.168.0.1

Edit hosts with ed? [no] <Entrée>

Do you want to do any manual network configuration? [no] <Entrée>
```

Il faut maintenant entrer le mot de passe root de la machine.

```
Password for root account? (will not echo) xxxxxxxx
```

```
Password for root account? (again) xxxxxxxx
```

Puis choisir le média d'installation...

```
Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk or (t)ape

device; or a (f)tp, (n)fs or (h)ttp server.

Where are the install sets? (or 'done') f

HTTP/FTP proxy URL? (e.g. 'http://proxy:8080', or 'none') [none] <Entrée>

Display the list of known ftp servers? [yes] n

Server? (IP address, hostname or 'done') 192.168.0.100

Does the server support passive mode ftp? [yes] <Entrée>

Server directory? [pub/OpenBSD/3.7/i386] pub/OpenBSD/i386

Login? [anonymous] <Entrée>
```

... et les ensembles de logiciels à installer. A nouveau pour faire simple, l'intégralité des packages sera installée :

```
The following sets are available. Enter a filename, 'all' to select

all the sets, or 'done'. You may de-select a set by prepending a '-'

to its name.


        [X] bsd

        [X] bsd.rd

        [ ] bsd.mp

        [X] base37.tgz

        [X] etc37.tgz

        [X] misc37.tgz

        [X] comp37.tgz

        [X] man37.tgz

        [X] game37.tgz

        [ ] xbase37.tgz

        [ ] xetc37.tgz

        [ ] xshare37.tgz
```

```
        [ ] xfont37.tgz
        [ ] xserv37.tgz
File Name? (or 'done') [bsd.rd] all

The following sets are available. Enter a filename, 'all' to select
all the sets, or 'done'. You may de-select a set by prepending a '-'
to its name.

        [X] bsd
        [X] bsd.rd
        [X] bsd.mp
        [X] base37.tgz
        [X] etc37.tgz
        [X] misc37.tgz
        [X] comp37.tgz
        [X] man37.tgz
        [X] game37.tgz
        [X] xbase37.tgz
        [X] xetc37.tgz
        [X] xshare37.tgz
        [X] xfont37.tgz
        [X] xserv37.tgz
File Name? (or 'done') [done] done
Ready to install sets? [yes] <Entrée>
```

L'installation des *files sets* de déroule. Il ne reste que quelques réglages (fuseau horaire, démarrage de certains services, ...) :

```
Where are the install sets? (or 'done') [done] done
Start sshd(8) by default? [yes] no
```

```
Start ntpd(8) by default? [no] <Entrée>

Do you expect to run the X Window System? [yes] <Entrée>

Change the default console to com0? [no] <Entrée>

Saving configuration files...done.

Generating initial host.random file...done.

What timezone are you in? ('?' for list) [Canada/Mountain] Europe/Paris
```

Il est souvent utile de remettre un MBR standard avec la commande **`fdisk -u wd0`**.

Configuration OpenBSD

Shell par défaut

Le shell par défaut sous OpenBSD est **csh**. Il est souvent plus agréable d'utiliser **ksh** pour ses fonctionnalités de rappel des commandes. Pour changer les paramètres système d'un utilisateur (nom, shell, répertoire de base), il faut utiliser la commande **`vipw`**.

Les services sous OpenBSD

Introduction

OpenBSD étant un système UNIX, les utilisateurs de Linux retrouveront sous OpenBSD de nombreuses notions.
Quelles sont les différences essentielles ?
Il n'y a pas de notions de runlevel. Les services sont simplement lancés ou non suivant les fichiers de configuration. A ce propos, on pourra noter par un **`ps auxw`** qu'il y a par défaut très peu de processus lancés.
Il y a également une notion propre à BSD qu'il est bon de souligner : le noyau peut élèver son niveau de sécurité (*securelevel*). Il empêche par exemple d'écrire dans /dev/mem ou /dev/kmem, il empêche le chargement de nouveau module, ... Cela a pour effet d'augmenter la sécurité et la stabilité du système. Surtout en cas de compromission du compte root !

la configuration

Les fichiers de configuration utilisés sont :

- **`/etc/rc`**

  : ce script est appelé par init, c'est lui qui lit la configuration, lance certains services et appelle les autres scripts
- **`/etc/rc.conf`**

  : contient la configuration de démarrage standard (inclut par

  `rc`

)
- **`/etc/rc.conf.local`**

: contient la configuration de démarrage personnalisée (inclut par **rc.conf**)

- **/etc/netstart** : ce script configure le réseau sur la machine (appelé par **rc**)
- **/etc/rc.securelevel** : ce scripts lance les services qui ont besoin de démarrer avant que le noyau n'élève son niveau de sécurité (appelé par **rc**)
- **/etc/rc.local** : script lancé en fin de démarrage, il sert à lancer les services non fournis dans la distribution standard (appelé par **rc**)

Voici trois extraits de **rc** :

```
# pick up option configuration

. /etc/rc.conf

if [ X${inetd} = X"YES" -a -e /etc/inetd.conf ]; then

        echo -n ' inetd';               inetd

fi



if [ X"${sshd_flags}" != X"NO" ]; then

        echo -n ' sshd';                /usr/sbin/sshd ${sshd_flags};

fi

[ -f /etc/rc.local ] && . /etc/rc.local
```

Voci deux extraits de **rc.conf** :

```
sshd_flags=""           # for normal use: ""

inetd=YES               # almost always needed

local_rcconf="/etc/rc.conf.local"



[ -f ${local_rcconf} ] && . ${local_rcconf} # Do not edit this line
```

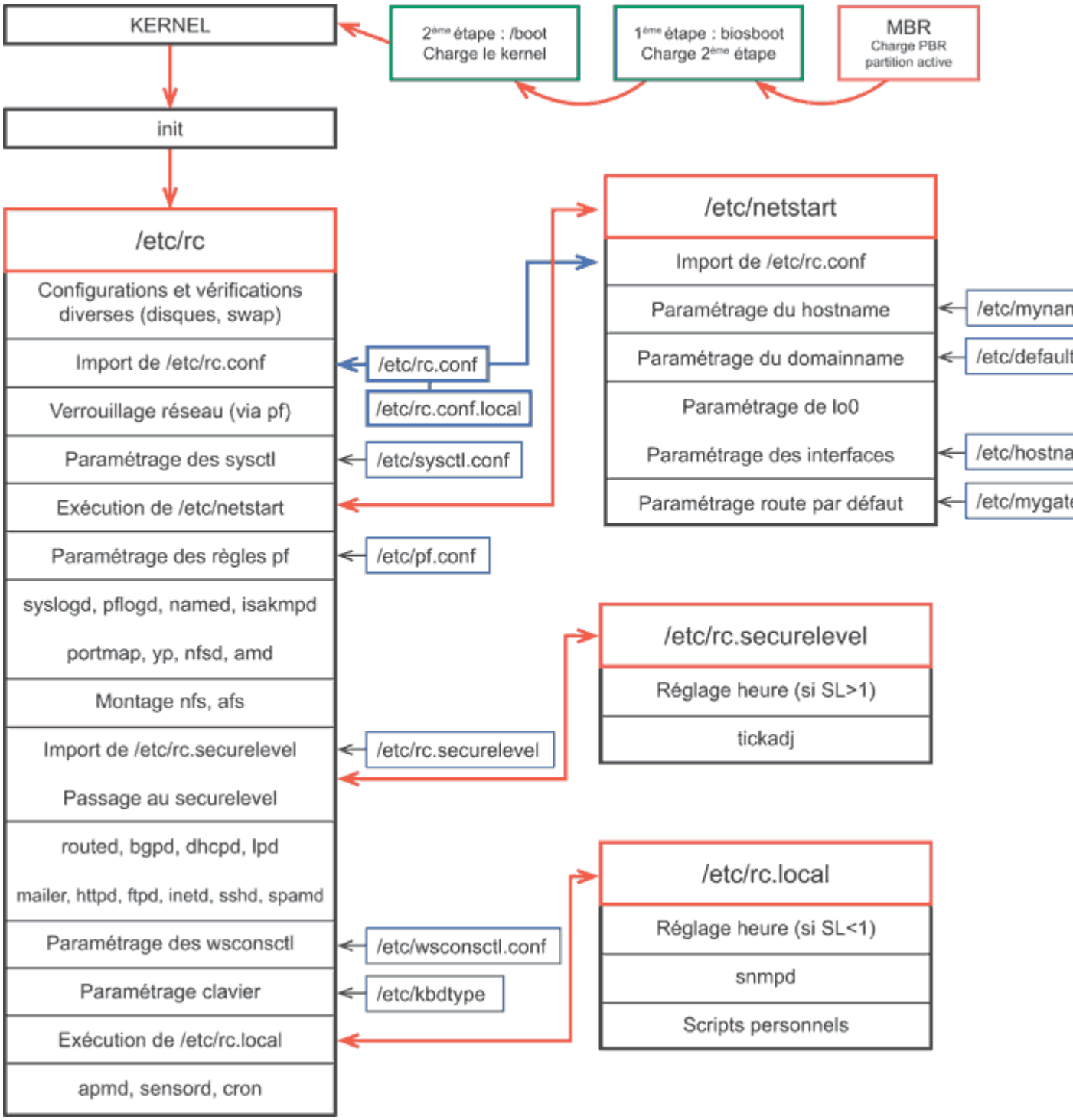
Le script **rc** est chargé de lancer les services au démarrage de la machine. Les services sont lancés ou non suivant les variables définies dans le fichier **rc.conf**.

Il est fortement recommandé de ne pas modifier les fichiers **rc** et **rc.conf**.

Si l'on souhaite lancer les services en plus, il faut ajouter le script de démarrage dans le fichier **rc.local**.

Si l'on souhaite changer les variables définies dans **rc.conf**, il faut les re-définir dans le fichier **rc.conf.local**.

Pour être tout à fait exact, le schéma suivant décrit le processus de démarrage d'OpenBSD. On peut y retrouver les appels aux fichiers de configuration précédemment cités.



Filtrage IP sur FWE

Cours sur IPtables et Netfilter

Netfilter est un filtre de paquets intégré dans le noyau Linux. Il est composé de tables qui indiquent les traitements qui doivent être appliqués sur les paquets passant par les interfaces réseau. La commande "iptables" sert à manipuler ces tables et donc à configurer Netfilter.

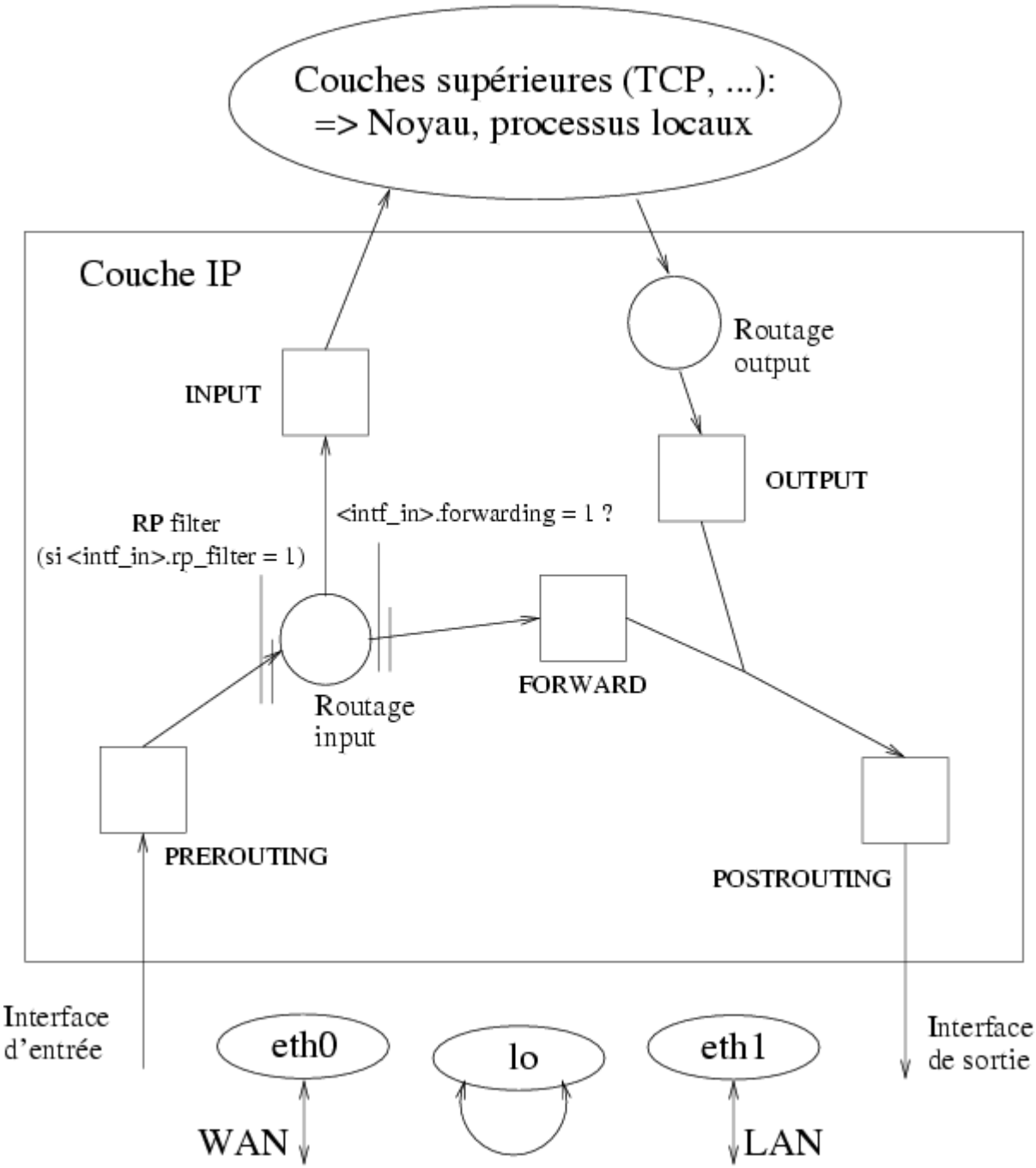
Netfilter permet essentiellement d'effectuer deux types de traitements différents :

- le filtrage, géré par la table filter (table par défaut), qui permet d'accepter ou de rejeter des paquets en fonction de la politique de sécurité de la machine
- la translation d'adresses, (NAT) qui permet de modifier l'adresse source ou destination des paquets, généralement pour partager des adresses IP entre plusieurs machines, et qui est gérée par la table nat.

Netfilter possède par ailleurs cinq points d'ancrage (hooks) au niveau de la pile IP :

- PREROUTING, qui intercepte tous les paquets entrants par les interfaces ;
- INPUT, en coupure des paquets destinés à la machine locale ;
- FORWARD, en coupure du trafic à router d'une interface vers une autre ;
- OUTPUT, en coupure des paquets émis depuis la machine locale ;
- POSTROUTING, qui intercepte tous les paquets sur le point de sortir par une interface.

Le schéma suivant illustre le positionnement des cinq points d'ancrage au sein de la couche IP du noyau Linux, en faisant également apparaître les points où le noyau consulte ses tables de routage pour orienter le paquet :



Le forwarding IP

On notera que le routage des paquets entrants permet de décider si les paquets sont destinés à la machine locale ou s'ils doivent être réémis.

Attention: un paquet sera vu comme destiné à la machine locale si son adresse destination est l'une quelconque de celles possédées par la machine, même s'il s'agit d'une adresse associée à une interface autre que celle par laquelle le paquet est arrivé.

Par défaut, une machine Linux ne route pas le trafic réseau qui ne lui est pas destiné. Ce comportement est contrôlé par une variable interne du noyau, que l'on peut modifier de deux façons différentes mais équivalentes :

- **`echo "1" > /proc/sys/net/ipv4/ip_forward`**

- **`sysctl -w net.ipv4.ip_forward=1`**

Le fichier de configuration **`/etc/sysctl.conf`** permet de paramétrer au démarrage les sysctl.

Le Reverse Path Filter

Un autre paramètre intéressant associé au routage est le "Reverse Path Filter". Lorsqu'il est activé, tous les paquets entrants (routables ou destinés à la machine locale) subissent une vérification d'adresse source. Si la table de routage indique qu'une réponse à ce paquet ne serait pas émise par l'interface par laquelle il a été reçu, l'adresse source est considérée falsifiée et le paquet est jeté.

Le Reverse Path Filter s'active sur l'ensemble des interfaces par la commande **`sysctl -w net.ipv4.conf.all.rp_filter=1`**

Le positionnement des tables

La table **filter** possède trois chaînes par défaut, qui s'insèrent au niveau des points d'ancrage du même nom :

- INPUT permet de filtrer le trafic entrant destiné à la machine locale;
- OUTPUT permet de filtrer le trafic issu de la machine locale;
- FORWARD permet de filtrer le trafic routé par le firewall.

La table **nat** possède au moins deux chaînes par défaut (ou plus selon les options de compilation du noyau), qui s'insèrent également au niveau des points d'ancrage correspondants :

- PREROUTING, qui permet d'éliminer la NAT sur les paquets entrants (utile pour la DNAT),
- POSTROUTING, qui permet de NATer les paquets sortants (utile pour la SNAT).

Le positionnement des chaînes de NAT permet donc d'écrire les règles de routage et de filtrage en utilisant les adresses non NATées (réelles).

Note : Il existe également une troisième table, **mangle**, destinée à permettre certaines manipulations sur les paquets (modification du TOS, du TTL, etc.). Cette table dispose de chaînes par défaut sur les cinq points d'ancrage.

Le fichier de base

Netfilter se configure via la commande "iptables" qui manipule les chaînes de règles dans les tables. On peut donc réaliser un script de configuration (script shell ou Makefile) qui appelle cette commande pour mettre en place les différentes règles. Une fois les règles chargées dans le noyau, il est possible d'automatiser leur rechargement en cas de redémarrage en utilisant une de ces deux commandes :

- **`/etc/init.d/iptables save`**
- **`iptables-save > /etc/sysconfig/iptables`**

Le script doit commencer par une réinitialisation de toutes les tables, afin de s'abstraire de la configuration particulière de Netfilter avant exécution du script. Il devrait donc commencer ainsi :

```
#!/bin/sh

IPTABLES=/sbin/iptables



$IPTABLES -P INPUT DROP
```

```
$IPTABLES -P OUTPUT DROP

$IPTABLES -P FORWARD DROP



$IPTABLES -F

$IPTABLES -t nat -F



$IPTABLES -X

$IPTABLES -t nat -X
```

Les trois premières lignes permettent de fixer la politique de filtrage par défaut pour les trois types de trafic : INPUT, OUTPUT et FORWARD. La politique par défaut est de jeter le paquet (DROP) si aucune règle ultérieure ne l'autorise explicitement. On peut remplacer DROP par ACCEPT si l'on souhaite que les paquets soient acceptés par défaut. Ce choix n'est pas raisonnable pour un firewall.

Les lignes suivantes permettent de vider les chaînes par défaut (-F) puis de supprimer les éventuelles chaînes utilisateur (-X).

Note : Cette remise à zéro ne doit pas créer d'état transitoire plus laxiste que la politique de sécurité en vigueur.

La mascarade

La mascarade est une translation d'adresses qui permet de partager, vis-à-vis de l'extérieur, une seule adresse IP externe (celle du firewall) entre plusieurs machines internes. On commence par créer des variables qui contiennent les paramètres réseau à utiliser pour écrire les règles. En voici un exemple :

```
IPTABLES="/sbin/iptables"      # exécutable iptables

IF_EXT="eth0"  # nom de l'interface externe du firewall

IF_INT="eth1"  # nom de l'interface interne du firewall

IP_EXT="192.168.0.1T"   # adresse externe (publique) du firewall

NET_DMZ="192.168.T.0/24"# adresses du réseau interne

NET_ANY="0.0.0.0/24"   # adresses quelconques

NET_WAN=$NET_ANY       # adresses du réseau externe
```

Dans cet exemple, l'interface réseau qui relie le firewall à l'Internet est l'interface eth0 et son adresse IP publique est 192.168.0.1T. Le firewall est relié à son réseau interne par l'interface eth1, et ce réseau utilise les adresses 192.168.T.0/24.

Afin d'effectuer la mascarade sur les connexions sortantes, le firewall va remplacer l'adresse source des paquets sortants par sa propre adresse IP externe (Source NAT : SNAT). Cette opération doit être faite après le routage (POSTROUTING). Elle correspond aux règles suivantes :

```
$IPTABLES -t nat -A POSTROUTING -o $IF_EXT -s $NET_DMZ -d $NET_WAN \

      -j SNAT --to-source $IP_EXT

$IPTABLES -A FORWARD -p tcp -i $IF_INT -o $IF_EXT -s $NET_DMZ -d $NET_WAN
--syn\

      -j ACCEPT

$IPTABLES -A FORWARD -p udp -i $IF_INT -o $IF_EXT -s $NET_DMZ -d $NET_WAN \

      -j ACCEPT

$IPTABLES -A FORWARD -p icmp -i $IF_INT -o $IF_EXT -s $NET_DMZ -d
$NET_WAN \

      -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED \

      -j ACCEPT
```

La première ligne est la règle de SNAT: elle indique que l'adresse source doit être remplacée par l'adresse IP externe du firewall sur tous les paquets sortants vers l'Internet. Cette règle ne concerne en fait que les premiers paquets des différentes communications, les communications établies étant gérées de façon transparente en POSTROUTING (modification de l'adresse source des paquets sortants) ou en PREROUTING (modification de l'adresse destination des paquets entrants pour assurer un routage vers le bon destinataire).
La règle de SNAT permet donc de NATer les connexions en provenance du réseau interne.

Les trois règles suivantes sont des règles de filtrage qui permettent de faire sortir (ACCEPT) du trafic spécifique (qui sera SNATé). La première règle indique que l'on peut laisser sortir toutes les demandes de connexion TCP (--syn). La deuxième règle indique que l'on peut laisser sortir tous les paquets UDP. Enfin, la troisième règle indique que l'on peut laisser sortir tous les paquets ICMP. On exprime le fait que les paquets soient sortants par les paramètres -i $IF_INT -o $IF_EXT -s $NET_DMZ -d $NET_WAN qui indiquent que les règles s'appliquent aux paquets issus du réseau interne et à destination du réseau externe.

Enfin, la dernière ligne indique que tous les paquets qui correspondent à une connexion établie (ESTABLISHED) sont autorisés à passer dans un sens ou dans l'autre. Netfilter étant un firewall stateful, il retient l'état de toutes les connexions associées aux flux qu'il voit passer. Il est donc capable de déterminer si un paquet correspond à une nouvelle connexion ou s'il poursuit une communication déjà établie (mêmes adresses, protocole, etc.). En plus du trafic identifié comme appartenant aux connexions établies, certaines nouvelles connexions sont considérées comme dérivées (RELATED) de connexions établies. C'est le cas des connexions de données FTP, ou encore des messages d'erreur ICMP. Ces connexions sont également autorisées par la dernière règle (le premier paquet est RELATED par rapport à la connexion d'origine et les suivants sont ESTABLISHED par rapport à la nouvelle).

Cette configuration fournit une connectivité quasi complète aux utilisateurs internes. Si l'on souhaite restreindre le trafic sortant, on peut limiter la portée des règles d'autorisation (sélection de ports, etc.) et/ou ajouter des règles de rejet explicite (-j DROP) avant les règles d'acceptation. Par exemple :

```
$IPTABLES -A FORWARD -o $IF_EXT -p tcp --dport 80 -j DROP
```

Cette règle permet de jeter (DROP) les connexions TCP sortantes sur le port 80 (http).

Remarques:

- Contrairement aux règles d'autorisation, qui doivent être les plus précises possibles, il n'est pas utile de restreindre outre mesure les règles d'interdiction tant qu'elles n'interfèrent pas avec le trafic autorisé.
- La règle d'autorisation générique pour les connexions établies et dérivées est trop laxiste en ce sens qu'elle ne permet aucune restriction et fait une confiance excessive au moteur de suivi d'états. Un traitement plus rigoureux (flux par flux) sera proposé dans la suite.

La DNAT

Le mécanisme de DNAT permet de fournir virtuellement les services de la DMZ au niveau du firewall externe (à son adresse publique) en redirigeant les requêtes vers la machine concernée. Dans une DMZ de production, les services sont normalement fournis par des machines différentes. Dans le cadre du TP, tous les services seront implantés sur la même machine : dmzT. On définit des variables pour identifier les adresses IP des machines et les ports des services :

```
IP_DMZ="192.168.T.2"

IP_SSH_SERVER=$IP_DMZ

IP_SMTP_SERVER=$IP_DMZ

IP_HTTP_SERVER=$IP_DMZ

IP_FTP_SERVER=$IP_DMZ

IP_DNS_SERVER=$IP_DMZ

PORT_SSH="22"

PORT_SMTP="25"

PORT_HTTP="80"

PORT_FTP="21"

PORT_DNS="53"
```

La redirection d'un service se fait alors au moyen de 2 règles : une règle de DNAT et une règle de filtrage pour autoriser le trafic correspondant. Dans le cas du service ssh cela donne :

```
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -s $NET_WAN -d $IP_EXT -p tcp
--syn \

      --dport $PORT_SSH -j DNAT --to-destination $IP_SSH_SERVER:$PORT_SSH

$IPTABLES -A FORWARD -i $IF_EXT -o $IF_INT -s $NET_WAN -d $IP_SSH_SERVER \
```

```
      -p tcp --syn --dport $PORT_SSH -j ACCEPT
```

Comme pour la SNAT, seul le premier paquet sera traité par la règle de NAT, le reste de la connexion étant géré de façon transparente par le suivi des états. En particulier, l'adresse source des réponses du serveur sera transformée en l'adresse publique du firewall (en POSTROUTING), à laquelle le client externe se croit connecté.

La règle de DNAT permet donc de NATer les connexions à destination de la DMZ.

Les services mail, http et ftp se traitent de façon similaire :

```
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -s $NET_WAN -d $IP_EXT -p tcp
--syn \

      --dport $PORT_SMTP -j DNAT --to-destination $IP_SMTP_SERVER:
$PORT_SMTP

$IPTABLES -A FORWARD -i $IF_EXT -o $IF_INT -s $NET_WAN -d $IP_SMTP_SERVER \

      -p tcp --syn --dport $PORT_SMTP -j ACCEPT



$IPTABLES -t nat -A PREROUTING -i $IF_EXT -s $NET_WAN -d $IP_EXT -p tcp
--syn \

      --dport $PORT_HTTP -j DNAT --to-destination $IP_HTTP_SERVER:
$PORT_HTTP

$IPTABLES -A FORWARD -i $IF_EXT -o $IF_INT -s $NET_WAN -d $IP_HTTP_SERVER \

      -p tcp --syn --dport $PORT_HTTP -j ACCEPT



$IPTABLES -t nat -A PREROUTING -i $IF_EXT -s $NET_WAN -d $IP_EXT -p tcp
--syn \

      --dport $PORT_FTP -j DNAT --to-destination $IP_FTP_SERVER:$PORT_FTP

$IPTABLES -A FORWARD -i $IF_EXT -o $IF_INT -s $NET_WAN -d $IP_FTP_SERVER \

      -p tcp --syn --dport $PORT_FTP -j ACCEPT
```

Pour les services UDP, les règles se construisent selon le même principe. Ainsi, pour le DNS :

```
$IPTABLES -t nat -A PREROUTING -i $IF_EXT -s $NET_WAN -d $IP_EXT -p udp \

      --dport $PORT_DNS -j DNAT --to-destination $IP_HTTP_SERVER:$PORT_DNS

$IPTABLES -A FORWARD -i $IF_EXT -o $IF_INT -s $NET_WAN -d $IP_DNS_SERVER \

      -p tcp --syn --dport $PORT_DNS -j ACCEPT
```

Note : Il est implicitement supposé dans ce qui précède que les paquets correspondant aux connexions établies ou dérivées ont été autorisées par la règle laxiste (cf. SNAT) :

```
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Construction de règles iptables

On remarque que le format d'écriture des règles est toujours le même:

- la commande iptables ;
- le choix de la table (-t table, ou filter par défaut) ;
- l'action et le choix de la chaîne (-A chaîne) ;
- les critères de sélection: sur les interfaces (-i intf_in, -o intf_out), les adresses (-s IP_src, -d IP_dst), le protocole (-p proto), les ports (--sport port_src, --dport port_dst), les flags TCP (--syn), l'état (-mstate --state état), etc ;
- la cible (-j cible) et ses éventuels arguments (pour la NAT).

La cible doit être adaptée à la table et à la chaîne choisies, ainsi -j SNAT n'est acceptable que pour la table nat et la chaîne POSTROUTING. De même, les critères de sélection disponibles dépendent du point d'ancrage considéré (-i intf_in n'a de sens que pour PREROUTING, INPUT et FORWARD, et -o intf_out pour FORWARD, OUTPUT et POSTROUTING).

Il est important d'insister sur le fait que les règles d'autorisation doivent être les plus strictes possibles et les règles d'interdiction les plus larges possible.

Affichage des règles actives

Pour afficher le contenu courant des tables et les statistiques d'utilisation des différentes règles, on peut utiliser les commandes **`iptables -L -v -n`**, **`iptables -t nat -L -v -n`** ou **`/etc/init.d/iptables status`**.

Exemple de configuration pour le FWE

Le fichier **`Makefile.fwe`** qui suit met en place un filtrage élémentaire sur le FWE. Ce filtrage, adapté au TP, peut encore largement être affiné avant usage en production.

```
############################

# Définition des variables #

############################

# Programmes exécutables

IPTABLES=/sbin/iptables

SYSCTL=/sbin/sysctl



# Interfaces locales

IF_EXT=eth0

IP_EXT=192.168.0.1T
```

```
IF_INT=eth1

IP_INT=192.168.T.1



# Autres machines

IP_DMZ=192.168.T.2

IP_FWI=192.168.T.3



# Sous-réseaux

NET_ANY=0.0.0.0/0

NET_WAN=${NET_ANY}

NET_TP=192.168.0.0/24

NET_DMZ=192.168.T.0/24



# Ports

PORT_TMP=1024:

PORT_ANY=:

PORT_FTP=21

PORT_SSH=22

PORT_ISAKMP=500

PORT_DNS=53

PORT_SMTP=25

PORT_HTTP=80



regular: clean generic ssh ftp dns smtp http regular-ipsec force-dnat
sysctl save



laxist: clean laxist-iptables sysctl save
```

```
clean:
	@echo
	@echo Politique par défaut pour le filtrage: DROP
	${IPTABLES} -P INPUT DROP
	${IPTABLES} -P OUTPUT DROP
	${IPTABLES} -P FORWARD DROP
	@echo
	@echo Politique par défaut pour la NAT: ACCEPT
	${IPTABLES} -t nat -P OUTPUT ACCEPT
	${IPTABLES} -t nat -P PREROUTING ACCEPT
	${IPTABLES} -t nat -P POSTROUTING ACCEPT
	@echo
	@echo Flushing des règles
	${IPTABLES} -F
	${IPTABLES} -t nat -F
	@echo
	@echo Suppression des chaînes utilisateur
	${IPTABLES} -X
	${IPTABLES} -t nat -X


save:
	@echo
	@echo Sauvegarde de la configuration courante iptables
	/etc/init.d/iptables save


sysctl:
	@echo
	@echo Activation du routage / IP Forwarding
```

```
	${SYSCTL} -w net.ipv4.ip_forward=1
	@echo
	@echo Activation anti-spoofing
	${SYSCTL} -w net.ipv4.conf.all.rp_filter=1

laxist-iptables:
	@echo
	@echo Politique par défaut pour le filtrage: ACCEPT
	${IPTABLES} -P INPUT ACCEPT
	${IPTABLES} -P OUTPUT ACCEPT
	${IPTABLES} -P FORWARD ACCEPT
	@echo
	@echo Installation de la SNAT
	${IPTABLES} -t nat -A POSTROUTING \
	   -o ${IF_EXT} -s ${NET_DMZ} -d ${NET_WAN} \
	   -j SNAT --to-source ${IP_EXT}

generic:
	@echo
	@echo Politique par défaut pour le filtrage: DROP
	${IPTABLES} -P INPUT DROP
	${IPTABLES} -P OUTPUT DROP
	${IPTABLES} -P FORWARD DROP
	@echo
	@echo Rejet des paquets etat INVALID
	${IPTABLES} -A FORWARD -m state --state INVALID -j DROP
	${IPTABLES} -A INPUT -m state --state INVALID -j DROP
	${IPTABLES} -A OUTPUT -m state --state INVALID -j DROP
```

```
@echo
@echo Autorisation connexions TCP sortantes
# N'inclut pas le FTP en mode actif (cnx serveur vers client)
${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
   -s ${NET_DMZ} -d ${NET_WAN} -p tcp \
   --sport ${PORT_TMP} --dport ${PORT_ANY} --syn \
   -j ACCEPT
${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
   -s ${NET_WAN} -d ${NET_DMZ} -p tcp \
   --sport ${PORT_ANY} --dport ${PORT_TMP} \
   -m state --state ESTABLISHED \
   -j ACCEPT
${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
   -s ${NET_DMZ} -d ${NET_WAN} -p tcp \
   --sport ${PORT_TMP} --dport ${PORT_ANY} \
   -m state --state ESTABLISHED \
   -j ACCEPT
@echo
@echo Autorisations connexions UDP sortantes
   ${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
   -s ${NET_DMZ} -d ${NET_WAN} -p udp \
   --sport ${PORT_TMP} --dport ${PORT_ANY} \
   -j ACCEPT
${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
   -s ${NET_WAN} -d ${NET_DMZ} -p udp \
   --sport ${PORT_ANY} --dport ${PORT_TMP} \
   -m state --state ESTABLISHED \
   -j ACCEPT
```

```
	@echo
	@echo Autorisation PING sortant
	${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
	   -s ${NET_DMZ} -d ${NET_WAN} -p icmp \
	   --icmp-type echo-request \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_WAN} -d ${NET_DMZ} -p icmp \
	   --icmp-type echo-reply \
	   -m state --state ESTABLISHED \
	   -j ACCEPT
	@echo
	@echo Autorisation ICMP erreur -- 2 sens
	${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
	   -s ${NET_DMZ} -d ${NET_WAN} -p icmp \
	   -m state --state RELATED \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_WAN} -d ${NET_DMZ} -p icmp \
	   -m state --state RELATED \
	   -j ACCEPT
	@echo
	@echo Installation de la SNAT
	${IPTABLES} -t nat -A POSTROUTING \
	   -o ${IF_EXT} -s ${NET_DMZ} -d ${NET_WAN} \
	   -j SNAT --to-source ${IP_EXT}


ssh:
```

```
	@echo
	@echo DNAT pour SSH
	${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} \
	   -s ${NET_TP} -d ${IP_EXT} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_SSH} \
	   -j DNAT --to-destination ${IP_DMZ}:${PORT_SSH}
	@echo
	@echo Autorisation connexions SSH entrantes
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_TP} -d ${IP_DMZ} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_SSH} --syn \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
	   -s ${IP_DMZ} -d ${NET_TP} -p tcp \
	   --sport ${PORT_SSH} --dport ${PORT_TMP} \
	   -m state --state ESTABLISHED \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_TP} -d ${IP_DMZ} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_SSH} \
	   -m state --state ESTABLISHED \
	   -j ACCEPT


ftp:
	@echo
	@echo DNAT pour FTP
	${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} \
	   -s ${NET_TP} -d ${IP_EXT} -p tcp \
```

```
        --sport ${PORT_TMP} --dport ${PORT_FTP} \
        -j DNAT --to-destination ${IP_DMZ}:${PORT_FTP}
    @echo
    @echo Autorisations pour le flux FTP de commandes
    ${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
        -s ${NET_TP} -d ${IP_DMZ} -p tcp \
        --sport ${PORT_TMP} --dport ${PORT_FTP} --syn \
        -j ACCEPT
    ${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
        -s ${IP_DMZ} -d ${NET_TP} -p tcp \
        --sport ${PORT_FTP} --dport ${PORT_TMP} \
        -m state --state ESTABLISHED \
        -j ACCEPT
    ${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
        -s ${NET_TP} -d ${IP_DMZ} -p tcp \
        --sport ${PORT_TMP} --dport ${PORT_FTP} \
        -m state --state ESTABLISHED \
        -j ACCEPT
    @echo
    @echo Autorisations pour les flux FTP de donnees PASV
    ${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
        -s ${NET_TP} -d ${IP_DMZ} -p tcp \
        --sport ${PORT_TMP} --dport ${PORT_TMP} --syn \
        -m state --state RELATED \
        -j ACCEPT
    # Les deux règles ESTABLISHED sont incluses dans les règles
    # d'autorisation des flux TCP sortants.
```

```
dns:
	@echo
	@echo DNAT pour DNS
	${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} \
	   -s ${NET_TP} -d ${IP_EXT} -p udp \
	   --sport ${PORT_TMP} --dport ${PORT_DNS} \
	   -j DNAT --to-destination ${IP_DMZ}:${PORT_DNS}
	@echo
	@echo Flux DNS
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_TP} -d ${IP_DMZ} -p udp \
	   --sport ${PORT_TMP} --dport ${PORT_DNS} \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
	   -s ${IP_DMZ} -d ${NET_TP} -p udp \
	   --sport ${PORT_DNS} --dport ${PORT_TMP} \
	   -m state --state ESTABLISHED \
	   -j ACCEPT


smtp:
	@echo
	@echo DNAT pour SMTP
	${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} \
	   -s ${NET_TP} -d ${IP_EXT} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_SMTP} \
	   -j DNAT --to-destination ${IP_DMZ}:${PORT_SMTP}
	@echo
	@echo Autorisation connexions SMTP entrantes
```

```
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_TP} -d ${IP_DMZ} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_SMTP} --syn \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
	   -s ${IP_DMZ} -d ${NET_TP} -p tcp \
	   --sport ${PORT_SMTP} --dport ${PORT_TMP} \
	   -m state --state ESTABLISHED \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_TP} -d ${IP_DMZ} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_SMTP} \
	   -m state --state ESTABLISHED \
	   -j ACCEPT

http:
	@echo
	@echo DNAT pour HTTP
	${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} \
	   -s ${NET_TP} -d ${IP_EXT} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_HTTP} \
	   -j DNAT --to-destination ${IP_DMZ}:${PORT_HTTP}
	@echo
	@echo Autorisation connexions HTTP entrantes
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_TP} -d ${IP_DMZ} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_HTTP} --syn \
	   -j ACCEPT
```

```
	${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
	   -s ${IP_DMZ} -d ${NET_TP} -p tcp \
	   --sport ${PORT_HTTP} --dport ${PORT_TMP} \
	   -m state --state ESTABLISHED \
	   -j ACCEPT
	${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \
	   -s ${NET_TP} -d ${IP_DMZ} -p tcp \
	   --sport ${PORT_TMP} --dport ${PORT_HTTP} \
	   -m state --state ESTABLISHED \
	   -j ACCEPT


force-dnat:
	@echo
	@echo Interdiction des cnx entrantes non DNATées
	${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} -j DROP


regular-ipsec:
	@echo
	@echo Flux ISAKMP
	# Sortie (SNAT faite par la règle générique)
	${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \
	   -s ${IP_FWI} -d ${NET_TP} -p udp \
	   --sport ${PORT_ISAKMP} --dport ${PORT_ISAKMP} \
	   -j ACCEPT
	# Entrée
	${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} \
	   -s ${NET_TP} -d ${IP_EXT} -p udp \
	   --sport ${PORT_ISAKMP} --dport ${PORT_ISAKMP} \
```

```
        -j DNAT --to-destination ${IP_FWI}:${PORT_ISAKMP}

    ${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \

        -s ${NET_TP} -d ${IP_FWI} -p udp \

        --sport ${PORT_ISAKMP} --dport ${PORT_ISAKMP} \

        -j ACCEPT

    @echo

    @echo Flux ESP

    # Sortie (SNAT faite par la règle générique)

        ${IPTABLES} -A FORWARD -i ${IF_INT} -o ${IF_EXT} \

        -s ${IP_FWI} -d ${NET_TP} -p esp \

        -j ACCEPT

    # Entrée

    ${IPTABLES} -A FORWARD -i ${IF_EXT} -o ${IF_INT} \

        -s ${NET_TP} -d ${IP_FWI} -p esp \

        -j ACCEPT

    ${IPTABLES} -t nat -A PREROUTING -i ${IF_EXT} \

        -s ${NET_TP} -d ${IP_EXT} -p esp \

        -j DNAT --to-destination ${IP_FWI}
```

Pour activer le filtrage correspondant à ce fichier, il suffit de le copier sur le FWE et de lancer la commande **`make -f Makefile.fwe`**.

Cours Cryptographie, Certificats IGC et OpenSSL

La Cryptographie

Nous ne donnons ici que les concepts minimaux et simplificateurs nécessaires à la réalisation du TP. La cryptographie est une science difficile et délicate. On pourra consulter, par exemple, un cours en ligne à l'adresse suivante : http://www.cacr.math.uwaterloo.ca/hac/

Le besoin de masquer les messages sensibles aux yeux des lecteurs indiscrets remonte à Jules César, voire avant. Aujourd'hui, on appelle cryptographie l'ensemble des connaissances et des techniques ayant pour but de rendre illisibles des messages aux observateurs autres que les destinataires légitimes. On parle de chiffrer car les messages sont assimilés à des nombres et les opérations de "masquage" de message sont, pour l'essentiel, des opérations mathématiques.

Le vocabulaire stabilisé en français est le suivant :

- On **chiffre** des messages ;
- On **déchiffre** des messages si on a la clé ;
- On décrypte des messages si on n'a pas la clé.

Tous les autres termes sont à proscrire, et en particulier le cryptage, l'encryptage, le chiffrage, la chiffration !

On distingue deux types d'algorithmes cryptographiques : les **algorithmes symétriques** qui utilisent la même clé pour chiffrer et déchiffrer les messages et les **algorithmes asymétriques** qui utilisent des bi-clés, une clé servant à chiffrer et l'autre à déchiffrer.

Les algorithmes symétriques sont rapides et performants au niveau de la qualité de la cryptographie. Malheureusement, ils souffrent d'un défaut majeur : les deux parties doivent partager une clé secrète pour pouvoir communiquer. Si une personne désire communiquer avec N autres personnes, plusieurs scénarios s'offrent à elle. Elle peut partager la même clé avec les N correspondants, mais ce n'est absolument pas satisfaisant, puisque si un de ses correspondants perd la clé, toutes les communications sont compromises. Elle peut également générer autant de clé que de correspondants, ce qui est fastidieux.
De plus, c'est là que le second défaut de la cryptographie symétrique apparaît : la transmission de la clé de chiffrement. Après avoir généré les clés de chiffrements, la personne doit les envoyer par des canaux différents de ceux qu'emprunteront ses communications (poste, téléphone), ce qui s'avère en pratique peu réalisable.
Exemple d'algorithmes symétriques : DES, TripleDes (ou 3DES), AES.

Les algorithmes asymétriques n'ont pas ce défaut : chaque utilisateur dispose d'une paire de clés (une clé de chiffrement, dite clé privée, et une clé de déchiffrement, dite clé publique). Il est impossible de recalculer une clé à partir de l'autre. Pour chiffrer un message, on utilise la clé publique. Seul le possesseur de la clé privée peut alors déchiffrer le message. Les algorithmes de cryptographie asymétrique sont basés sur des propriétés mathématiques complexes qu'il est inutile d'aborder ici. L'inconvénient de ces protocoles est qu'ils sont très coûteux en ressources et donc très lents.
Exemple d'algorithme asymétrique : RSA.

Les solutions performantes sont donc des solutions hybrides où la communication est chiffrée par une clé dite de session basée sur un protocole symétrique. Cette clé est générée aléatoirement et transmise via un protocole asymétrique.
Le protocole entre A et B a donc la forme suivante :

- A tire une clé de session k.
- B envoie sa clé publique à A.
- A chiffre k avec la clé publique de B et envoie le résultat à B.
- B le déchiffre avec sa clé privée.
- A et B communiquent en utilisant un protocole symétrique donc la clé est k.

Le problème de cette méthode est que rien ne garantit à A que c'est bien la clé publique de B qu'il reçoit. Imaginons qu'un pirate se place au milieu de cette communication, s'il envoie sa clé publique à la place de B, il prend sa place dans toute la suite de la communication : ce problème est appelé "man in the middle".

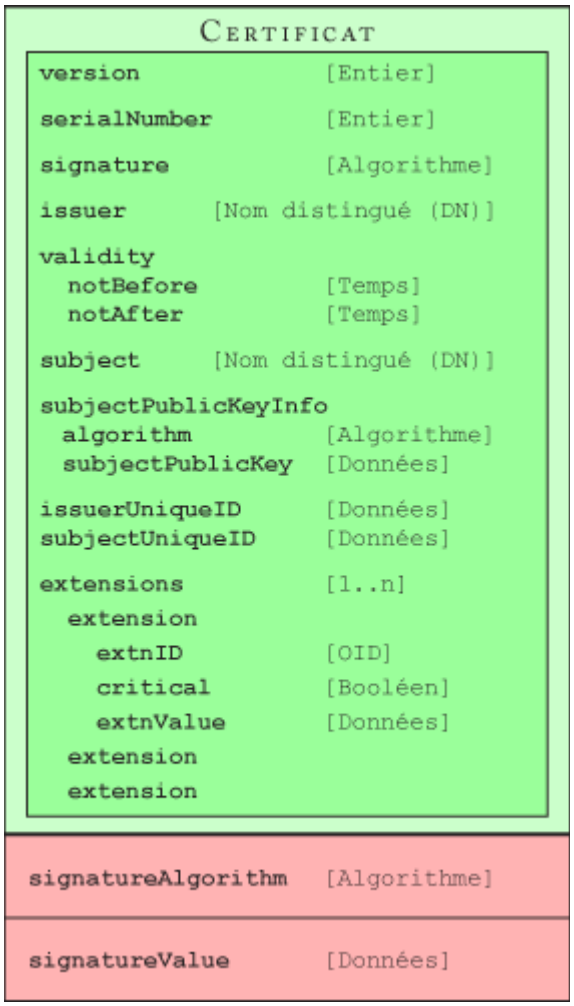
Il faut donc utiliser la signature numérique pour garantir que la clé publique de B est bien celle de B (authentification), et n'a pas été modifiée (intégrité). On se base alors sur des protocoles de signatures (par exemple ceux basés sur RSA ou DSA).

Le mécanisme de signature est le suivant : on souhaite envoyer un message signé. Le message à envoyer est hashé, c'est à dire que l'on va calculer une somme de contrôle du texte. Pour cela, on utilise des algorithmes comme md5 ou sha1 (de préférence), qui garantissent que si le texte change, ne fût-ce que d'un octet, le hashé (ou condensat) changera considérablement. Ensuite le condensat est utilisé avec la clé **privée** de l'émetteur et la fonction de signature pour générer la signature numérique. Le destinataire récupère le message et cette signature numérique. Il recalcule le condensat et le vérifie avec la signature reçue et la clé **publique** de l'émetteur. Si la fonction de vérification de la signature valide le condensat, c'est que l'émetteur du message est bien le propriétaire de la clé privée associée à la clé publique utilisée pour la vérification ce qui garantit aussi bien l'identité de l'envoyeur que l'intégrité du message, pour peu que l'on soit sûr d'avoir la clé publique du correspondant.

Car, là encore, rien ne prouve que la clé publique que l'on possède est bien celle du correspondant légitime.

Pour la signature numérique, on pourra consulter la formation en ligne à l'adresse : http://www.formation.ssi.gouv.fr/autoformation/signature.html

Les Certificats



Afin de pallier ce défaut, l'idée est d'utiliser des "objets" garantissant qu'une clé publique présentée appartient bien à son légitime propriétaire. Ce sont les certificats. En première approche, on peut considérer un certificat comme l'association d'une identité et d'une clé publique, le tout signé par la clé privée d'une autorité. Ces certificats pourront être vérifiés par la clé publique de cette autorité qui la diffuse le plus largement possible via un certificat.

Les certificats sont normalisés par la norme X509 qui définit des champs de bases (X509) ainsi que des extensions (X509v3). Le cadre d'utilisation est normalisé par la RFC 3280.

La structure d'un certificat est détaillée dans l'image de droite. Le détail des champs est le suivant :

- version : version de la norme X509 utilisée.
- serialNumber : numéro de série unique du certificat.
- signature : algorithme de signature utilisé dans la signature du certificat.
- issuer (émetteur) : nom distingué de l'autorité ayant signé ce certificat.
- validity : date de début et de fin de validité du certificat.
- subject (objet) : nom distingué du détenteur du certificat et de la clé privée associée.
- subjectPublicKeyInfo : clé publique contenu dans le certificat.
- extensions : extensions x509v3.

Les noms distingués sont des enregistrements stucturés au format x501. Ces informations peuvent être le nom courant (**CN** pour *Common Name*), l'organisation (**O**), l'unité d'organisation (**OU**), le pays (**C** pour *Country*), l'adresse e-mail (**E**), une description (**D**), ...

Les certificats ont également des extensions dont les plus utilisés sont :

- AuthorityKeyIdentifier : identifiant de la clé de l'autorité.
- SubjectKeyIdentifier : identifiant de la clé du sujet.
- KeyUsage : précise le cadre d'utilisation du bi-clé du certificat.
- SubjectAltName : non alternatif pour le sujet du certificat.
- BasicConstraints : identification pour un certificat d'une autorité.
- ExtendedKeyUsage : précise le cadre d'utilisation du certificat.
- CRLDistributionPoints : point de distribution de listes de révocations (CRL).

Rôles pour KeyUsage :

- digitalSignature : signature numérique (hors certificat ou de CRL).
- nonRepudiation : signature numérique et non répudiation.
- keyEncipherment : chiffrement de clé pour le transport, utilisé en particulier avec un bi-clé RSA.
- dataEncipherment : chiffrement de données (hors clé), utilisé en particulier avec un bi-clé RSA.
- keyAgreement : mise en accord de clés, utilisé en particulier avec Diffie-Hellman.
- keyCertSign : signature numérique de certificats, nécessaire pour les certificats d'autorité.
- cRLSign : signature numérique de liste de révocation, nécessaire pour les certificats d'autorité.
- encipherOnly : doit être utilisé avec keyAgreement.
- decipherOnly : doit être utilisé avec keyAgreement.

Exemples de rôle (liste non exhaustive) pour extendedKeyUsage :

- serverAuth : authentification d'un serveur dans le cadre d'une authentification TLS (SSL).
- clientAuth : authentification d'un client dans le cadre d'une authentification TLS (SSL).
- codeSigning : signature de code exécutable (.exe, ActiveX, applet Java).
- emailProtection : utilisation avec S/MIME dans une messagerie sécurisée.
- timeStamping : horodatage des données.
- OCSPSigning : signature de données OCSP.

Pour le TP, les combinaisons suivantes seront utilisées (définies dans le fichier **openssl.cnf**) :

```
# Extensions pour l'authentification d'un client (authentification TLS et
messagerie sécurisée)

keyUsage             = digitalSignature, nonRepudiation, keyEncipherment

extendedKeyUsage     = clientAuth, emailProtection
```

```
# Extensions pour l'authentification d'un serveur (authentification TLS)

keyUsage             = digitalSignature, keyEncipherment

extendedKeyUsage     = serverAuth
```

Ces certificats sont codés au format binaire ASN.1 : c'est le **format DER**. Les extensions généralement utilisées sont .cer ou .der.
Il existe aussi le **format PEM** utilisant le codage en base 64 avec des délimiteurs. L'extension généralement utilisée est .pem

Exemple de délimiteurs :

```
----BEGIN RSA PRIVATE KEY----

----END RSA PRIVATE KEY----



----BEGIN CERTIFICATE----

----END CERTIFICATE----



----BEGIN CERTIFICATE REQUEST----

----END CERTIFICATE REQUEST----



----BEGIN X509 CRL----

----END X509 CRL----
```

Enfin, il existe un autre format, le **PKCS#12**, qui est un codage binaire incluant un certificat, sa clé privée associée et optionnellement d'autres certificats. Le fichier peut être chiffré à l'aide d'une passphrase. Ce format est utilisé par les IGC pour distribuer les certificats et les clés privées à leurs clients. L'extension utilisée est .p12 ou .pfx

Les Infrastructures à Gestions de Clés (IGC)

Une autorité est donc un organisme qui va donner à ses utilisateurs des certificats qui permettent de les identifier. Ces certificats pourront être vérifiés par la clé publique de cette autorité qui diffuse le plus largement possible son propre certificat.

À propos de l'autorité, il faut distinguer plusieurs entités, même si dans le cadre du TP, elles sont confondues. Chaque autorité a un rôle précis. L'**autorité d'enregistrement** est l'entité qui enregistre la demande de certificat. Elle est chargée de vérifier l'identité du demandeur ainsi que la

correspondance entre la clé publique présentée et sa clé privée. Ce rôle est donc très important puisque c'est sur ces vérifications que repose la confiance que l'on accorde à l'autorité. L'**autorité de certification** délivre le certificat en signant la requête présentée par l'autorité d'enregistrement. Il est évident que la clé privée de cette autorité, celle qui signe les certificats, doit être conservée de manière extrêmement sécurisée puisque c'est sur ce secret que repose toute l'infrastructure technique de l'IGC. Enfin, les **services de publications** mettent les certificats à la disposition des utilisateurs ; ils doivent également publier les listes de révocations contenant une liste de certificats invalidés pour diverses raisons (vol ou perte de clé privée en général). Ces trois services sont en général assumés par la même entité, appelée autorité de confiance. C'est sur la base de cette autorité que l'on construit une IGC.

La création d'une IGC commence par la création du certificat de cette autorité. Pour cela, on commence par générér un bi-clé et l'identité de l'IGC. Le certificat de l'IGC est ensuite signé avec la clé privée associée à la clé publique contenue dans le certificat. Ce certificat est donc auto-signé. Ensuite, l'autorité doit diffuser largement et de façon fiable ce certificat notamment en l'intégrant dans les outils qui l'utiliseront.

Une fois que l'autorité est créée, son travail consiste à fournir des certificats à ses clients. Pour cela, on procède en deux temps.
On commence par générer un bi-clé et une identité pour le client. La clé publique et l'identité sont contenus dans une requête. Cette requête doit être présentée à l'autorité pour être signée.
Dans un deuxième temps, l'autorité signe cette requête (en y ajoutant des extensions). On obtient alors le certificat de l'utilisateur (l'association entre son identité et sa clé publique) signé avec la clé privée de l'autorité. Il ne reste plus qu'à délivrer au client son certificat et la clé privée associé.

Ainsi, lors d'une communication sécurisée, les interlocuteurs n'échangeront pas leur clé publique mais leur certificat. La vérification de ce certificat avec la clé publique du certificat de l'autorité (largement diffusé) garanti la non falsification des certificats. Les utilisateurs devront alors prouver qu'ils possèdent la clé privée associée au certificat présenté.

OpenSSL

OpenSSL est un package logiciel incluant une librairie assez complète de cryptographie, un programme (openssl) qui permet d'utiliser la plupart des fonctionnalités de la librairie et un **script CA** pour automatiser les traitements liès à la gestion d'une IGC. Les paramètres des IGC sont configurés via un fichier nommé **openssl.cnf**.

Voici quelques exemples d'utilisation de cette boîte à outils.
La syntaxe est la suivante : **openssl < type de traitement > [arguments spécifiques ...]**
Le premier argument spécifie le type d'opération que l'on souhaite effectuer (génération d'une clé, hashage d'un document, ...), puis suivent les arguments spécifiques à l'opération demandée. Les pages de manuel sont très bien documentées.

```
#Génération d'une bi-clé RSA de 1024 bits protégée par 3DES :

openssl genrsa -out clersa.pem -des3 1024



#Génération d'une requête de certificat au format PEM pour la clé générée :
```

```
openssl req -new -key clersa.pem -outform PEM -out requete.pem



#Chiffrement symétrique d'un document en AES 128 avec une clé dérivée d'un
mot de passe :

openssl enc -aes-128-cbc -in texte_clair.txt -out texte_chiffre.enc -salt
-e -k MotDePasse



#Affichage textuel des données d'un certificat :

openssl x509 -in certificat.pem -text
```

<<< Précédent
Cours Cryptographie, Certificats IGC et OpenSSL

Suivant >>>
Installation et configuration des services (inetd)

Sommaire

Mise en place d'une IGC

Détails de l'IGC

Dans ce TP, on souhaite mettre en place une IGC à deux niveaux.
Le professeur dispose d'une autorité racine et chaque groupe dispose d'une autorité intermédiaire, signée par l'autorité racine du professeur. Cette autorité délivrera des certificats pour le groupe, en l'occurrence un certificat pour un utilisateur (pour une utilisation avec S/MIME et pour l'authentification client TLS) et des certificats pour les services (pour l'authentification serveur TLS).

Mise en place de l'IGC

Le professeur dispose de son autorité racine, c'est-à-dire qu'il a généré un certificat autosigné. Comme précisé dans le cours précédent, ce sont les extensions x509v3 qui détermine le rôle du certificat. En l'occurence, ce certificat auto-signé peut servir à générer d'autres certificats, il a donc les extensions **`keyUsage = keyCertSign, cRLSign`** et **`basicConstraints = critical,CA:TRUE`**.

Pour toutes les opérations qui suivent, on utilise le script **`CA`**. En effet, ce script automatise et met en forme les appels à la commande openssl, ce qui simplifie la syntaxe des commandes à taper.

Configuration

Récupérer le script **`CA`** et le fichier de configuration **`openssl.cnf`** **depuis le serveur**. Les placer dans votre répertoire **`/root/`**.

Il faut ensuite les modifier, notamment pour spécifier les répertoires de travail et nommer correctement les sections.

Pour le fichier **openssl.cnf** :

```
[ ca ]

default_ca = CA_GROUPET



[ CA_GROUPET ]

dir = /root/CA_Base
```

Pour le fichier **CA** il faut créer (si elle n'existe pas) le variable d'environnement SSLEAY_CONFIG spécifiant le chemin du fichier de configuration openssl.cnf :

```
SSLEAY_CONFIG="-config /root/openssl.cnf"

DAYS="-days 3650"

CATOP=/root/CA_BASE
```

Création d'une autorité

Chaque stagiaire doit commencer par générer les fichiers de son autorité :

```
cd /root

./CA -newca
```

Voici les opérations réalisées par le script :

- il commence par créer l'arborescence CA_Base (**/root/CA_Base/**) ;
- il génère une clé RSA, demande un mot de passe pour la protéger. La clé privée est stockée dans le fichier **/root/CA_Base/private/cakey.pem** ;
- il faut ensuite entrer tous les renseignements caractérisant l'autorité (nom, localisation géographique, ...). Le script crée alors le certificat de l'autorité dans le fichier **/root/CA_Base/cacert.pem**.

Création d'un certificat serveur

Chaque stagiaire peut maintenant générer ses certificats.

Pour générer un certificat servant à l'authentification serveur, il convient de modifier quelques lignes dans le fichier **/root/ca/openssl.cnf** afin de préciser les extensions x509v3 concernant l'usage du certificat. Par exemple, pour un certificat d'authentification d'un serveur avec TLS :

```
keyUsage          = digitalSignature, keyEncipherment

extendedKeyUsage  = serverAuth
```

Note : mettre en commentaire les autres lignes keyUsage et extendedKeyUsage

Il reste à générer le certificat. On commence par générer un bi-clé et préparer une requête avec la commande **./CA -newreq**. Les renseignements d'identification concernant l'entité propriétaire du futur certificat sont demandés. La requête se trouve dans le fichier **newreq.pem** et la clé privée dans le fichier **newreq.pem** .

Il faut maintenant signer la requête avec la commande **./CA -sign**. Il faut entrer le mot de passe protégeant la clé privée de l'autorité afin de pouvoir l'utiliser pour signer la requête. La requête signée devient un certificat et se trouve dans le fichier **newcert.pem**.

Création d'un certificat client

Éditer le fichier openssl.cnf pour choisir les extensions à ajouter au certificat.

```
keyUsage          = digitalSignature, keyEncipherment, nonRepudiation

extendedKeyUsage  = clientAuth, emailProtection
```

Note : commenter les autres lignes keyUsage et extendedKeyUsage.

On effectue alors la même procédure que pour le certificat serveur :

- **./CA -newreq** : génération d'un bi-clé et de la requête (stockés dans les fichiers **newkey.pem** et **newreq.pem**) ;
- **./CA -sign** : signature de la requête. Le certificat se trouve dans le fichier **newcert.pem** ;

Création d'un fichier au format PKCS#12

Le format PKCS#12 définit un format de fichier permettant de stocker un certificat d'un utilisateur, sa clé privée associée et éventuellement d'autres certificats. Afin de protéger la clé privée, le fichier est protégée à l'aide d'un mot de passe.

La commande suivante permet de générer le fichier PKCS#12 (il faut se placer dans le répertoire **/root/**) :

```
openssl pkcs12 -export -in newcert.pem -inkey newkey.pem -certfile
CA_Base/cacert.pem -out user.p12
```

Note : il ne peut n'y avoir qu'un seul paramètre -certfile. S'il faut intégrer plusieurs certificats, il faut les concaténer dans un seul fichier.

Maintenance de la base de l'IGC

Révocation d'un certificat

Pour révoquer un certificat, il faut disposer de celui-ci. Tous les certificats signés par l'IGC étant copiés dans le dossier **/root/CA_Base/newcerts**, il suffit de ce placer dans ce répertoire et de lancer la commande **openssl ca -config /root/openssl.cnf -revoke cert.pem**, où cert.pem désigne le certificat que l'on désire révoquer. Le fichier index.txt est alors mis à jour.

Génération de la liste de révocation

```
openssl ca -config /root/openssl.cnf -gencrl -out crl.pem -crldays 30
```

Pour voir le contenu d'un fichier CRL, il faut utiliser la commande suivante : **`openssl crl -in crl.pem -text`**

Mise à jour de la base

Il faut marquer comme expiré les certificats dont les dates de validités ne sont plus valables. La commande suivante permet de mettre à jour automatiquement tous les certificats périmés du fichier index.txt : **`openssl ca -config /root/openssl.cnf -updatedb`**

Intégration des certificats clients

Microsoft Windows

Récupérer le fichier user.p12 et double-cliquer dessus. Les données stockées dans le fichier PKCS#12 étant protégées, il faut donner le mot de passe. Cliquer sur suivant. Sélectionner automatiquement le magasin : cliquer sur oui. Le certificat de l'utilisateur la clé privée sont installés.

Installation et configuration des services sur la DMZ

La phase suivante est l'installation des services les plus classiques offerts par Internet. Pour des raisons pratiques, ils sont tous installés sur la même machine. Il est évident qu'en production, il faut appliquer au maximum la règle "d'unicité des fonctions" (un seul service sur chaque machine). Ceci pour éviter que la compromission d'un service (le serveur web, par exemple) entraîne l'arrêt en cascade d'autres services (mail, accès internet, vpn, ...).
On remarquera cependant qu'à défaut d'isolation physique, un soin particulier est apporté à l'isolation locale des processus par l'emploi de chroot et de comptes utilisateurs dédiés.

Le super serveur Internet : inetd

Inetd est un "super-serveur" Internet. Le super-serveur est un programme qui écoute les connexions réseau et les redirige vers le programme approprié.

La configuration de inetd est assez simple, elle se trouve dans le fichier **`/etc/inetd.conf`**. Ce fichier contient toute la configuration du super-serveur. Il est organisé sous forme de lignes. Chaque ligne désigne un service. Voyons un extrait de ce fichier :

```
ftp stream tcp nowait root /usr/libexec/ftpd -l -a
```

- "ftp" est le nom du service tel qu'on le trouve dans **`/etc/services`** (ce qui indique pour TCP/UDP le numéro de port)
- "stream" indique le mode connecté, par opposition à "dgram"
- "tcp" indique le protocole de transport utilisé (voir le fichier **`/etc/protocols`**)
- "nowait" indique que le serveur ftp est multi-threadé (le serveur n'est pas bloqué à chaque nouvelle connexion)
- "root" est l'utilisateur sous lequel tourne le serveur ftp
- "/usr/libexec/ftpd" est le nom du serveur dédié, suivi des options de ses démarrage.

La plupart des services sont prévus dans le fichier. Pour les activer, il suffit de supprimer le # en début de ligne et de relancer inetd. Dans le cas de la DMZ, il faut penser à désactiver tous les services inutiles (time, ident, ...)

Si un nouveau service doit être lancé, il suffit d'ajouter une ligne similaire à celle présentée.

Il faut également noter l'existence, sous forme de packages, de xinetd. C'est une version plus modulaire du super-serveur, en général utilisée sous Linux.

Installation et configuration des services sur la DMZ

Le serveur web Apache

Sous OpenBSD, Apache, intégré dans le système de base, est chrooté et tourne sous un compte utilisateur non privilégié (www en l'occurrence).

Le chroot est un mécanisme Unix qui permet d'enfermer dans « une prison » un programme potentiellement dangereux. Les binaires enfermés dans cette prison voient le répertoire prison (typiquement **`/var/www/`** pour le serveur web Apache) comme la racine de l'arborescence (**`/`**) et ne peuvent théoriquement pas en sortir. Cela permet d'éviter que la compromission d'un service n'entraîne la compromission de tout le système.

Il faut avoir conscience que les deux mécanismes de sécurité mis en oeuvre dans OpenBSD (chroot et setuid) sont complémentaires : chrooter un service qui tourne en root est absolument inutile car root peut facilement sortir de sa prison.

Cette sécurité par défaut offerte par OpenBSD est très pratique car mettre en place ces mécanismes à la main est assez fastidieux (il faut copier beaucoup de choses dans la prison d'Apache : l'exécutable, ses fichiers de configuration, toutes ses librairies et tous ses modules)

La configuration par défaut fournie par OpenBSD est presque prête à l'emploi : il suffit de modifier quelques lignes dans le fichier de configuration principal pour que le service fonctionne correctement.

Editer le fichier **`/var/www/conf/httpd.conf`** et modifier les paramètres suivants :

```
ServerRoot "/var/www"ServerAdmin webmaster@groupeT.tpServerName
www.groupeT.tpDocumentRoot "/var/www/htdocs"
```

Commandes de base

Quelques commandes de base utiles :

- **`httpd -t`**

  : test de la configuration du serveur
- **`apachectl start`**

  : Démarrage du serveur
- **`apachectl stop`**

  : Arrêt du serveur

- **`apachectl restart`**

  : Redémarrage du serveur

Modules d'apache

Apache possède une structure modulaire (via d'API DSO). Des modules peuvent être rajoutés afin d'étendre les fonctionnalités du serveur. Par défaut, apache (sous entendu le binaire httpd) possède déjà un certain nombre de modules intégrés au binaire httpd. La commande **`httpd -l`** permet de lister ces modules intégrés.

Il est important de n'utiliser que les modules nécessaires. Certains modules sont indispensables au fonctionnement d'Apache, d'autre peuvent être enlevés sans aucun problème.

Voir la page http://httpd.apache.org/docs/mod/index-bytype.html pour la liste des modules de base ou notre **tableau synthétique**.

Par exemple, les modules suivants sont considérés comme sensibles, il est préférable de les enlever.

```
mod_status   : affiche l'etat du systeme dans une page HTML

mod_info     : affiche des informations sur la configuration du serveur

mod_cgi      : execution des script CGI (HTML dynamique)

mod_userdir  : repertoires personnels d'utilisateurs

langages specifiques :

perl_module

php3_module

php4_module

dav_module

python_module

put_module
```

Trois directives de configuration permettent de manipuler les modules :

- **`LoadModule <chemin_du_module>`** : permet de charger un module et de l'activer
- **`ClearModuleList`** : efface toute la liste des modules actifs, y compris ceux integrés dans l'exécutable Apache.
- **`AddModule <nom du module>`** : active un module

On commentera les lignes associées aux modules dangereux.

Il faut également noter que l'ordre de chargement des modules est très important puisque certains modules dépendent d'autres.

Configuration divers

Le paramètre **Servertokens** spécifie la chaîne d'identification du serveur. On a le choix entre **Prod Min**, **OS**, **Full**. On choisira le moins verbeux, c'est à dire **Prod**.

Le paramètre **ServerSignature** spécifie le format de la ligne d'information dans certaines pages autogénérées. On a le choix entre **On**, **Off** et **EMail**. On choisira le moins verbeux, c'est à dire **Off**.

Modifier le fichier **httpd.conf** et régler ces deux paramètres :

```
Servertokens Prod

ServerSignature Off
```

Intégration du certificat dans Apache

Pour rajouter le support de TLS dans Apache, il faut générer un certificat avec le rôle d'authentification server (serverAuth).

Il est recommandé de mettre les certificats et surtout la clé privée en dehors de la cage d'Apache (**/var/www/**). Le propriètaire de ces fichiers doit être l'utilisateur root et les permissions en lecture uniquement pour root, c'est à dire -r--------.

Il faut ensuite modifier quelques lignes dans le fichier **/var/www/conf/httpd.conf** pour activer la couche TLS d'Apache et l'utiliser avec le certificat :

```
SSLCertificateFile /etc/ssl/apache_cert.pem     # Fichier contenant le
certificat

SSLCertificateKeyFile /etc/ssl/apache_cle.pem   # Fichier contenant la clé
privée associée au certificat

SSLCACertificatePath /etc/ssl/CA.pem            # Certificat de l'autorité
ayant délivré les certificats
```

Installation et configuration des services sur la DMZ

Le service de nom (DNS) : Bind

Sous OpenBSD 3.7, le service DNS (Bind 9.3.0) est inclus dans le système de base. Il est configuré par défaut pour se chrooter (dans **/var/named**) et tourner sous un utilisateur non privilégié (named).

La seule tâche est donc la configuration de notre serveur DNS. Récupérer les fichiers **named.conf** et **groupeT.tp** sur le ftp. Placer **named.conf** dans **/var/named/etc/** puis placer **groupeT.tp** dans **/var/named/master/**. Il y a deux fichiers de configuration à éditer.

Configuration du serveur DNS

Le fichier **/var/named/etc/named.conf** est découpé en plusieurs sections (liste partielle, voir le *BIND 9 Administrator Reference Manual* pour plus de détails) :

- acl
- controls
- logging
- options
- server
- view
- zone

La section **acl** permet de définit les listes d'accès réutilisables dans d'autres sections du fichier de configuration. La définition suivante permet de définir les clients internes :

```
acl clients_interne {

        localhost;

        localnets;

        192.168.1T.0/24;

};
```

La section **controls** définit les paramètres pour l'administration à distance via l'utilitaire **rndc**. Pour désactiver cette fonctionnalité, il ne faut spécifier aucun paramètre :

```
controls { };
```

La section **logging** permet de définir la journalisation des requêtes. Laisser les paramètres par défaut :

```
logging {

        category lame-servers { null; };

};
```

La section **options** définit les paramètres généraux du serveur et des sections de zone qui définissent le comportement du serveur pour les requêtes DNS.

```
options {

        version "";                     // information de version :
désactivation



        listen-on    { 127.0.0.1; 192.168.T.2; }; // interfaces d'écoutes
IPv4

        listen-on-v6 { none; };         // interfaces d'écoutes IPv6 :
désactivation
```

```
        allow-recursion { none; };      // désactivation des requêtes
recursives

        allow-transfer { none; };       // désactivation du transfert de
zone



        notify no;                      // désactivation des notifications
de changement de zones

        zone-statistics no;             // désactivation des statistiques

        interface-interval 0;           // désactivation de la recherche de
nouvelles interfaces

        max-cache-size 20M;             // taille max du cache

};
```

Les sections **view** définissent des comportements du serveur sur la base de l'adresse IP du client qui envoie la requête, permettant de différencier les réponses DNS. On définit ainsi deux vues :

- une correspondant aux clients de la zone interne et DMZ : il faut réactiver la récursion pour ces requêtes, ainsi que permettre de résoudre tous les noms possible (zone ".").
- une autre correspondant aux requêtes en provenance de l'extérieur (par exemple Internet). Il faut autoriser le requêtes que pour la zone ou le serveur DNS est l'autorité, c'est à dire la zone **groupeT.tp**.

```
view "interne" {

        match-clients   { clients_interne; };

        recursion       yes;

        allow-recursion { clients_interne; };



        // racine

        // type hint correspond aux requêtes que l'on ne peut pas traiter
localement

        // le fichier root.zone doit être récupéré régulièrement sur
ftp.rs.internic.net/domain

        zone "." {

                type    hint;

                file    "standard/root.hint";

        };
```

```
        zone "localhost" {
                type    master;
                file    "standard/localhost";
                allow-transfer { localhost; };
        };

        zone "127.in-addr.arpa" {
                type    master;
                file    "standard/loopback";
                allow-transfer { localhost; };
        };

        zone "com" {
                type delegation-only;
        };

        zone "net" {
                type delegation-only;
        };

        // zone de type master correspond aux requêtes locales : ce serveur
est l'autorité
        // de cette zone (groupeT.tp)
        zone "groupeT.tp" {
                type    master;
                file    "master/int.groupeT.tp";
        };
};
```

```
view "externe" {
        match-clients   { any; };
        recursion       no;

        // zone de type master correspond aux requêtes locales : ce serveur
est l'autorité
        // de cette zone (groupeT.tp)
        zone "groupeT.tp" {
                type    master;
                file    "master/ext.groupeT.tp";
        };
};
```

Configuration d'une zone

Voici un exemple de fichier de zone. Celui-ci correspond à la zone **"groupeT.tp"** pour la vue interne. Les adresses données en réponse sont donc les adresses IP internes.

Fichier **`/var/named/master/int.groupeT.tp`** :

```
$TTL 86400      ; durée de vie par défaut des enregistrements en secondes

@       IN SOA  dmzT.groupeT.tp. root.groupeT.tp. (
                2002090601      ; numéro de serie de la zone
                3H              ; refresh
                15M             ; retry
                1W              ; expiry
                1D )            ; minimum

        IN NS           dmzT.groupeT.tp.        ; nameserver gérant la zone
        IN MX   10      mail                    ; serveur de mail de la
zone
```

```
dmzT    IN A            192.168.T.2             ; définition d'une machine
mail    IN CNAME        dmzT                    ; définition d'un alias
www     IN CNAME        dmzT
```

Note : par convention, le serial est de la forme AAAAMMJJRR (AAAA : année, MM : mois, JJ : jour, RR : révision).

Post-Installation

Lancement manuel du serveur : **`/usr/sbin/named`**

A chaque modification des fichiers de zones, on peut les recharger sans arrêter le serveur avec la commande **`kill -HUP `cat /var/run/named.pid``**, à condition d'incrémenter le numéro de série de la zone (paramètre **`serial`**).

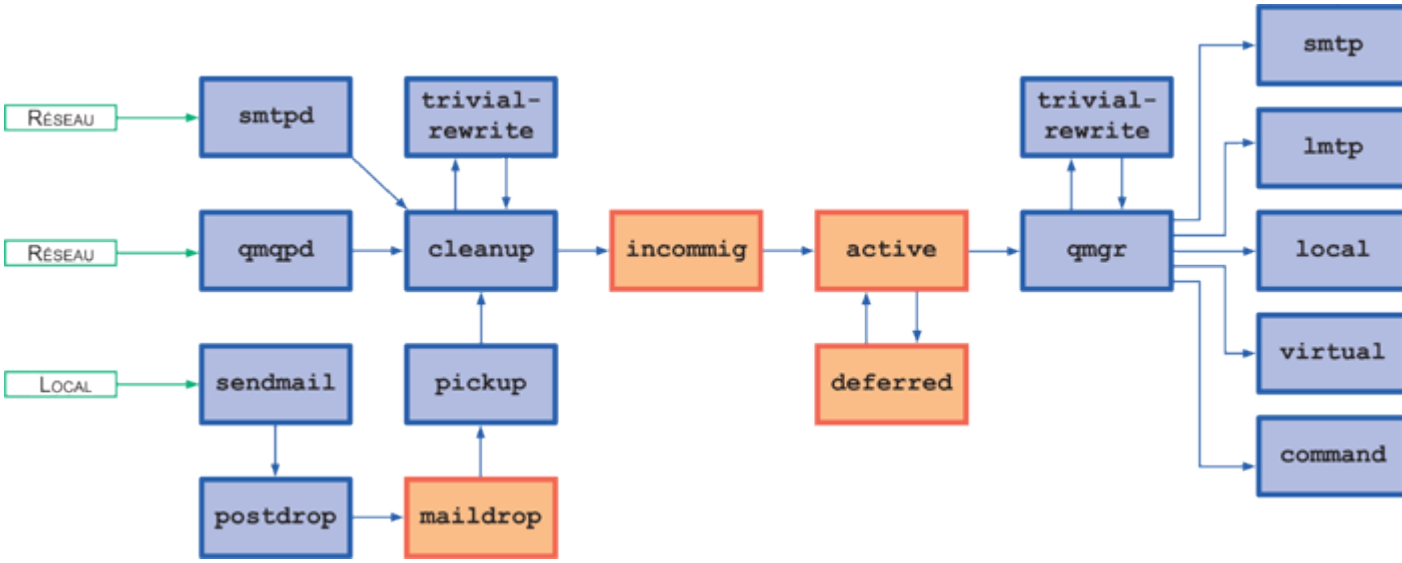
Installation et configuration des services sur la DMZ

Le relais mail : Postfix

Il faut maintenant installer un service des plus importants sur Internet : le mail.

Postfix est un relais mail (MTA pour Mail Transfert Agent). OpenBSD est fourni en standard avec Sendmail (celui-ci étant configuré pour une utilisation locale uniquement). Le choix s'est porté sur Postfix car il est plus simple à configurer et axé sur la sécurité.

Le schéma suivant décrit l'architecture interne de postfix.



Remarques sur ce schéma :

- les rectangles bleus représentent les processus

- les rectangles rouges représentent les files d'attente
- les processus représentés dans le cadre sont contrôlés par le processus master
- les données représentées dans le grand cadre sont la propriété de Postfix

Installation

Sur la machine dmz, installer avec le package Postfix avec support TLS : **`pkg_add postfix-2.0.18.tls0.8.16-tls.tgz`**.

Après l'installation, il faut encore activer postfix à la place de Sendmail. Il suffit de suivre les instructions données à la fin de l'installation de Postfix. Notamment, il faut supprimer les tâches cron de sendmail et configurer le système pour qu'il utilise postfix à la place de Sendmail.

C'est ce que fait le script **`postfix_enable`**

Postfix ne pratique pas la descente de privilège (setuid) comme Bind ou Apache mais plutôt la séparation de privilège. C'est-à-dire que chaque processus ne tourne qu'avec les privilèges dont il a strictement besoin (paticulièrement les processus qui écoutent le réseau). Mais certains processus ont besoin des privilèges root (en particulier le processus master).

Configuration

Les fichiers de configuration sont dans le répertoire **`/etc/postfix`**.

Récupérer le fichier **`main.cf`** sur le serveur ftp et remplacer le fichier existant (**`/etc/postfix/main.cf`**). Il s'agit du fichier d'origine sans les commentaires. Ce fichier liste les principaux paramètres de configuration.

Identification de la machine :

```
myhostname = mail.groupeT.tp

mydomain = groupeT.tp

myorigin = $mydomain
```

Liste des domaines et utilisateurs gérés par le serveur :

```
mydestination = $myhostname, localhost.$mydomain, $mydomain

local_recipient_maps = $alias_maps, proxy:unix:passwd.byname
```

Liste des réseaux autorisés à relayer via ce serveur :

```
mynetworks = 127.0.0.1/32, 192.168.1T.0/24

relay_domains = $mydestination
```

Paramètres divers :

```
smtpd_banner = $HOSTNAME ESMTP

setgid_group = postdrop

mail_owner = postfix
```

```
inet_interfaces = all

queue_directory = /var/spool/postfix

notify_classes = resource, software, bounce
```

Base des alias :

```
alias_maps = hash:/etc/mail/aliases

alias_database = hash:/etc/mail/aliases
```

Le fichier **/etc/postfix/master.cf** spécifie les processus et les options associées utilisés par postfix. Désactiver les processus non utilisés tels que cyrus, uucp, ifmail, bsmtp.

Dans le fichier **/etc/mail/aliases**, ajouter l'alias _postfix pointant vers root :

```
_postfix: root
```

Construire la base de donnée des alias avec la commande **newaliases** ou **postalias hash:/etc/mail/aliases**.

Lancement

Postfix se lance avec la commande **postfix start**.

Regarder dans les fichiers **/var/log/messages** et **/var/log/maillog** pour voir si tout s'est bien passé. Si ce n'est pas le cas, modifier la configuration et relancer le serveur avec les commandes **postfix stop** puis **postfix start**.

Filtres

Il est possible de mettre en place des filtres. Par exemple pour les entêtes, on peut supprimer des entêtes indésirables ou rejeter le mail. Commencer par spécifier dans le fichier **/etc/postfix/main.cf** le nom du fichier traitant les entêtes.

```
header_checks = regexp:/etc/postfix/header_checks
```

Créer ensuite ce fichier de vérification des entêtes, **/etc/postfix/header_checks** :

```
/^Subject: Make Money Fast/  REJECT

/^To: friend@public.com/     REJECT



/^X-MS/            IGNORE

/^X-Mailer/        IGNORE

/^X-Receive/       IGNORE
```

Les lignes se présentent sous la forme :

```
/Expression régulière/      ACTION
```

Les lignes REJECT indiquent que si le mail contient l'entête qui matche l'expression régulière, le mail ne sera pas délivré.
Les lignes IGNORE indiquent que si le mail contient un entête qui matche l'expression régulière, l'entête sera supprimée mais le mail sera délivré.

Dans cet exemple, les deux première lignes effectuent un filtrage (très) basique anti-spam.
Les trois dernières sont plus intéressantes : elles sont là pour éviter la fuite d'information. En effet, dans un mail, certains champs optionnels peuvent contenir des informations d'architecture interne : adresse IP du client ou du relais mail interne, nature du mailer utilisé par le client, ... Il est préférable de supprimer ces champs qui, en outre, n'ont pas d'utilité propre dans le protocole SMTP.

Commandes diverses

Arrêt et démarrage de Postfix, vérification de la configuration :

- **`postfix check`** : vérification élémentaire de la configuration
- **`postfix reload`** : rechargement des fichiers de configuration
- **`postfix start`** : démarrage de postfix
- **`postfix stop`** : arrêt de postfix

Gestion des files d'attente :

- **`mailq`** : affichage du contenu des files
- **`postqueue -p`** : affichage du contenu des files
- **`postqueue -f`** : forcer le traiment des files
- **`postfix flush`** : forcer le traiment des files

La commande **`postmap`** permet de mettre à jour les bases (access, canonical, virtual, relocated, tranport) sauf la base des aliases.

```
postmap hash:/etc/postfix/virtual
```

Pour la base des aliases, il faut utiliser la commande **`postalias`** ou **`newaliases`**. La commande newaliases traite le fichier spécifié dans **`main.cf`** (paramètre **`alias_database`**).

```
postalias hash:/etc/postfix/aliases

newaliases
```

Pour forcer la prise en compte immédiate des bases utiliser la commande **`postfix reload`**.

La commande postconf permet de voir la configuration de postfix.

- **`postconf`** : paramètres en cours
- **`postconf -d`** : paramètres par défault
- **`postconf -n`** : paramètres différant de ceux par défaut

Gestion des files d'attentes :

- **`postcat (-v)`**
- **`postsuper -d All`** : suppression de tous les messages
- **`postsuper -d MESSAGEID`** : suppression d'un message

Points de vérification

- La version du serveur est t'elle à jour ?
    - Commande postconf mail_version (affiche la version de postfix)
- Les processus tournent t'ils avec un utilisateur non privilégié ?
    - Flag unprivileged du fichier /etc/postfix/master.cf
    - Commande ps aux
- Le serveur est-il chrooté ?
    - Flag chroot du fichier /etc/postfix/master.cf
    - Utilisation de la commande fstat
- Quels sont les domaines autorisés à relayer via ce serveur ?
    - Paramètre smtpd_recipient_restrictions du fichier /etc/postfix/main.cf
- Chaîne d'identification du serveur
    - Paramètre smtpd_banner de main.cf
    - Commande telnet localhost 25

Installation et configuration des services sur la DMZ

Un serveur imap : imap-uw

Le service IMAP est beaucoup plus complet que POP. Il offre la possibilité aux utilisateurs de créer des répertoires dans leurs boite mail. D'autre part, les messages restent sur le serveur, ce qui permet de lire ou de récupérer ses mails depuis plusieurs systèmes différents.

Installation

Après avoir installé un relais mail, il reste à installer un serveur imap qui permet de récupérer ses mails. imap-uw (de l'University of Washington) a été choisi pour sa simplicité de configuration (il n'y a pas de fichier de configuration).

Installer le package d'imap-uw : **`pkg_add imap-uw-2002.339-plaintext.tgz`**.

Il existe deux packages pour imap-uw : **imap-uw-2002.339-plaintext.tgz** et **imap-uw-2002.339.tgz**. Le package imap-uw-2002.339-plaintext.tgz autorise l'authentification simple (c'est à dire sous la forme login et mot de passe) en clair sur port 143, alors que le package imap-uw-2002.339.tgz ne permet l'authentification simple uniquement sur un canal TLS (port 993).

Ajouter ces deux lignes dans **`/etc/inetd.conf`** :

```
imap stream tcp nowait root /usr/local/libexec/imapd imapd

pop3 stream tcp nowait root /usr/local/libexec/ipop3d ipop3d
```

Redémarrer inetd avec la commande **`pkill -HUP inetd`**.

Il est possible de véridier si le serveur fonctionne correctement en tapant sur la dmz : **`telnet localhost 143`**.

Le serveur doir répondre par une ligne du type :

```
+OK IMAP Welcome to GNU IMAP 0.9.9-2 <22464.1037799490@dmzT>
```

Utiliser la command LOGOUT dans le telnet pour quitter la session.

Ajout du support de TLS

Après avoir essayé ce service en clair, on peut installer le package imposant l'utilisation de TLS pour les authentifications en clair. Il faut d'abord cependant supprimer le package installé.

```
pkg_del imap-uw-2002.339-plaintext
pkg_add imap-uw-2002.339-tls.tgz
```

Pour rajouter le support de TLS dans imap-uw, il faut générer un certificat avec le rôle d'authentification serveur (serverAuth). Placer le certificat et la clé dans le même fichier (**/etc/ssl/imapd.pem**).

Modifier les deux lignes dans **/etc/inetd.conf** pour écouter sur les ports utilisant TLS :

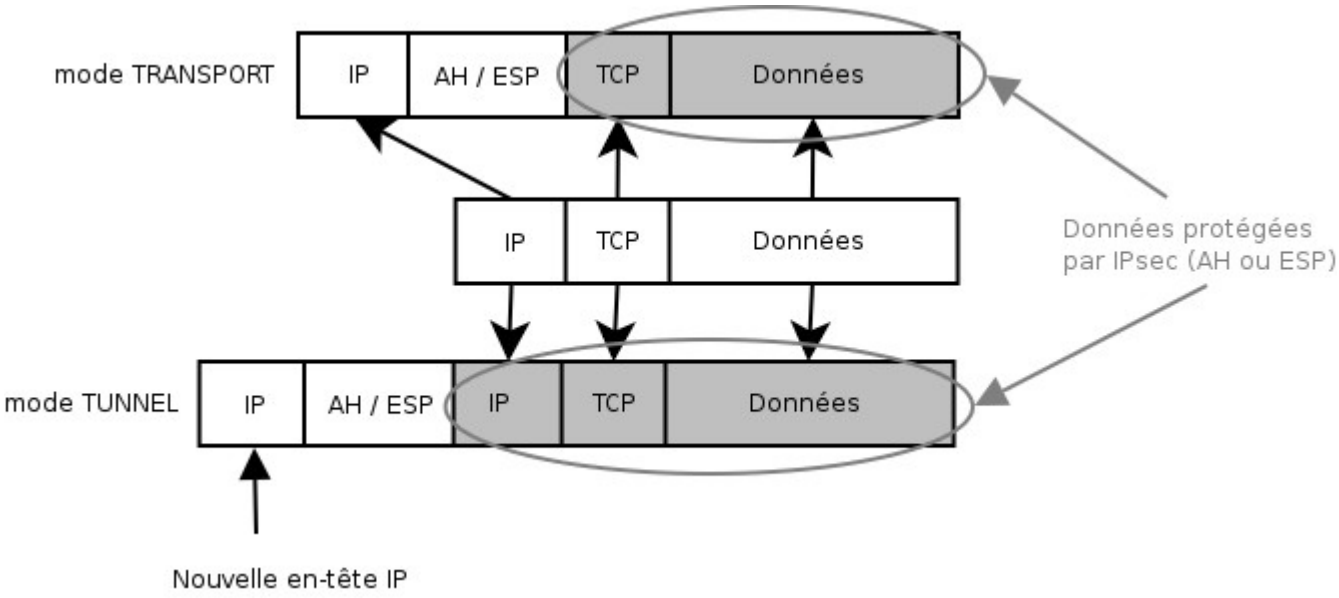
```
imaps stream tcp nowait root /usr/local/libexec/imapd imapd

pop3s stream tcp nowait root /usr/local/libexec/ipop3d ipop3d
```

Mise en place d'un Réseau Privé Virtuel (vpn)

Aspect réseau : les protocoles utilisés

IPsec est une extension du protocole IP (couche 3 du modèle TCP/IP), incluse nativement dans IPv6 mais optionnelle dans IPv4. Son utilisation est transparente pour les protocoles des couches supérieures, donc pour les utilisateurs et les applications. Une communication IPsec peut fonctionner suivant deux modes différents : le mode transport et le mode tunnel.

Le mode transport offre essentiellement une protection aux protocoles de niveau supérieur (TCP, UDP) mais ne modifie pas la partie IP (les adresses IP sources et destination restent inchangées); il peut être utile dans le cas d'une communication sécurisée au sein d'un LAN, par exemple. Le second permet quant à lui d'encapsuler la totalité des datagrammes IP dans d'autres datagrammes IP, dont le contenu est protégé. L'intérêt majeur de ce second mode est qu'il rend possible la mise en place de passerelles de sécurité entre deux réseaux, ces passerelles chiffrant ou déchiffrant symétriquement l'ensemble des flux IP concernés.



Le protocole IPsec utilise deux sous-protocoles pour la protection des datagrammes : AH et ESP. AH permet l'authentification, le contrôle d'intégrité et l'anti-rejeu. ESP offre les mêmes fonctionnalités plus la confidentialité.

AH (authentication header)

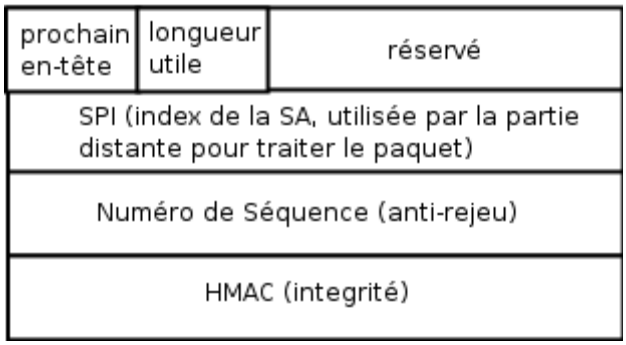
C'est le premier des protocoles de protection des données de la spécification IPsec. Il est détaillé dans la RFC 2402.
Il garantit :

- L'authenticité des datagrammes IP reçus.
- L'intégrité des champs suivants du datagramme IP : données (en mode tunnel, ceci comprend la totalité des champs, y compris les en-têtes, du datagramme IP encapsulé dans le datagramme protégé par AH), version (4 en IPv4, 6 en IPv6), longueur de l'en-tête (en IPv4), longueur totale du datagramme (en IPv4), longueur des données (en IPv6), identification, protocole ou en-tête suivant (ce champ vaut 51 pour indiquer qu'il s'agit du protocole AH), adresse IP de l'émetteur, adresse IP du destinataire (sans source routing). AH assure donc l'intégrité des champs IP qui ne sont pas susceptibles d'être modifiés pendant le routage, les autres champs (TTL, etc.) ne pouvant être contrôlés.
- L'unicité (optionnelle, au choix du récepteur), ce qui empêche le rejeu.

Le protocole AH n'assure pas la confidentialité : les données sont authentifiées mais pas chiffrées.

Enfin, AH ne spécifie pas d'algorithme particulier. Ceux-ci sont décrits séparément, ce qui fait la souplesse du protocole IPsec. Cependant, une implémentation conforme à la RFC 2402 est tenue de supporter les algorithmes MD5 et SHA-1, les algorithmes de hachage les plus courants.

en-tête AH



ESP (encapsulating security payload)

ESP est le second protocole de protection des données de la spécification IPsec. Il est détaillé dans la RFC 2406. Contrairement à AH, ESP ne protège que les données des datagrammes IP, pas les en-têtes. Ce protocole est en général plus utilisé que AH puisqu'il inclut des mécanismes de chiffrement.

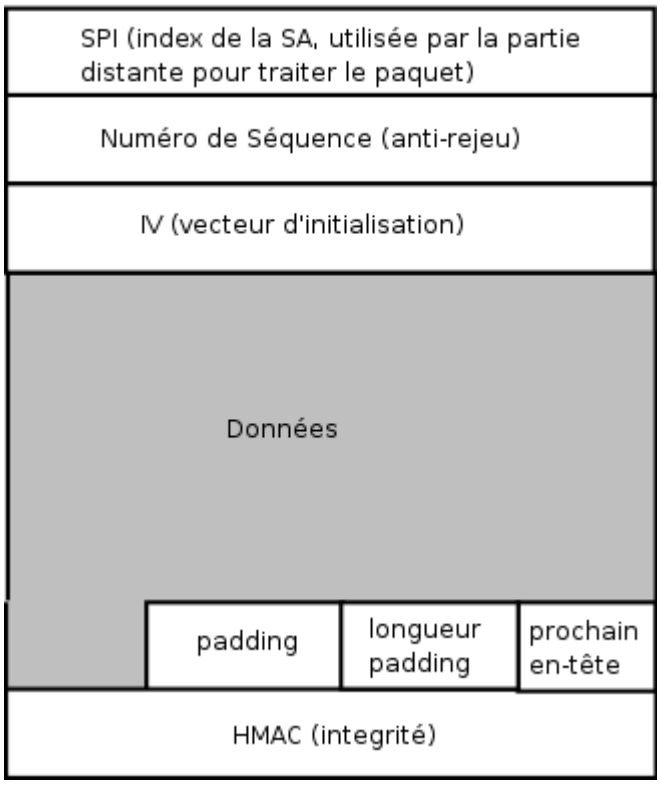
En mode transport, il assure :

- La confidentialité des données (optionnelle) : la partie données des datagrammes IP transmis est chiffrée.
- L'authenticité des datagrammes IP reçus (optionnelle, mais obligatoire en l'absence de confidentialité). Il faut noter que, si l'authentification d'origine n'est pas imposée par le protocole, elle est en pratique quasi-indispensable dans la mesure où elle contribue à la protection contre les attaques actives.
- L'intégrité de la partie données du datagramme IP (si les datagrammes sont authentifiés).
- L'unicité (optionnelle, au choix du récepteur), ce qui empêche le rejeu.

En mode tunnel, ces garanties s'appliquent aux données du datagramme dans lequel est encapsulé le trafic utile, c'est-à-dire à la totalité (en-tête et options inclus) du datagramme encapsulé.
Ce mode assure donc une meilleure confidentialité sur les flux de données : un attaquant externe qui capture le trafic encapsulé ne peut pas identifier l'émetteur ou le destinataire des datagrammes.

Enfin, comme AH, ESP ne spécifie pas d'algorithme particulier. Ceux-ci sont décrits séparément. Cependant, une implémentation conforme à la RFC 2406 est tenue de supporter l'algorithme de chiffrement DES en mode CBC et les algorithmes de hachage MD5 et SHA-1.

en-tête et suffixe ESP



Dans l'exemple du TP, on utilise ESP en mode tunnel pour que le VPN soit le plus souple et le plus robuste possible.

IKE (internet key exchange)

Dans la plupart des cas, les clés utilisées pour chiffrer les données d'un tunnel ne sont pas fixées avant sa mise en place.
Une gestion manuelle des clés de chaque tunnel est d'autant plus fastidieuse qu'il y a de chiffreurs. De plus, la robustesse des algorithmes cryptographiques n'est garantie que si les clés sont renouvelées régulièrement (gestion de l'usure et de la cryptopériode). Enfin, la compromission d'un équipement permet, si la clé peut être récupérée (par exemple sur le disque), de déchiffrer à posteriori tout le trafic chiffré s'il a été enregistré.

Le protocole IKE permet une gestion dynamique des clés IPsec reposant sur une architecture de confiance (PKI/certificats X509, secret pré-partagé, etc.). IKE réalise une négociation des protocoles et des algorithmes à utiliser pour chaque tunnel, ainsi que les échanges de clés nécessaires à leur mise en oeuvre dans la durée (incluant les renouvellements). Ces échanges sont protégés par une authentification reposant sur l'architecture de confiance retenue. Une compromission à posteriori des éléments de cette architecture ne permet que des attaques actives sur le trafic futur. L'utilisation du mode PFS (*Perfect Forward Secrecy*) permet de plus de rendre les différentes clés générées indépendantes les unes des autres.

Le protocole IKE, qui est assez complexe, est décrit en détail dans la RFC 2409.

Aspect système : fonctionnement d'un élément VPN

La gestion locale d'IPsec au niveau d'un hôte ou d'une passerelle repose sur deux types d'objets : les SA et les SP.

SA : Security Association

Une association de sécurité caractérise un demi-canal IPsec entre deux entités, c'est-à-dire : une source, une destination, un sens de communication, des protocoles utilisés (pour l'authentification, le chiffrement, la compression), des algorithmes cryptographiques, leurs paramètres et les clés utilisées. Chaque SA est repérée par un SPI (security parameter index). La base des SA valides au niveau d'une entité IPsec est la SAD (Security Association Database).

SP : Security Policy

Une politique de sécurité IPsec est une règle indiquant, pour un type particulier de flux, quel traitement doit être appliqué: autorisation simple (choix par défaut), interdiction, ou traitement IPsec. Les critères utilisables pour sélectionner un flux sont multiples: adresses, protocoles, ports, etc. Les politiques indiquant un traitement IPsec renvoient vers les SA disponibles en spécifiant les caractéristiques requises. La base des SP en vigueur au niveau d'une entité IPsec est la SPD (Security Policy Database).

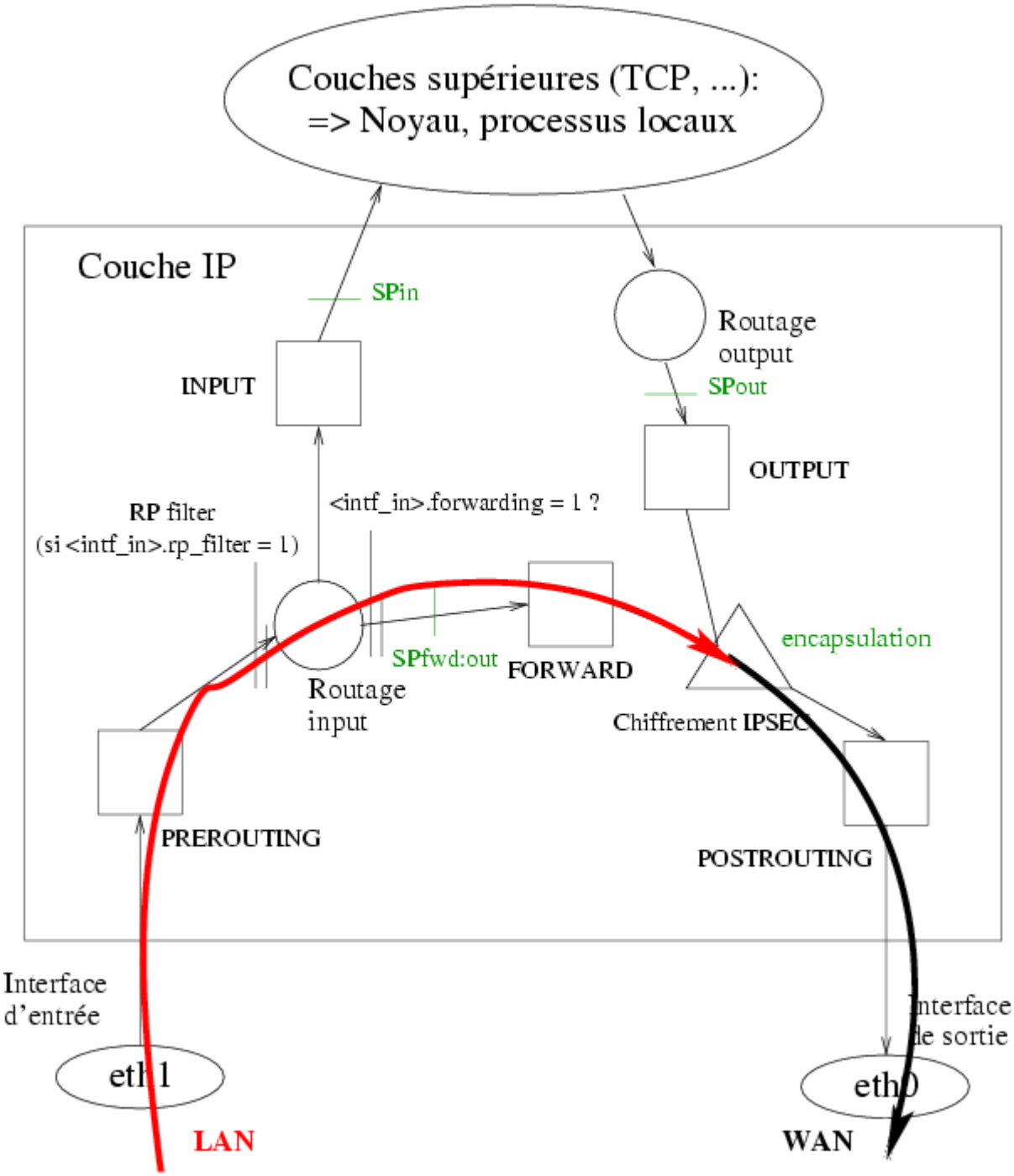
La SPD définit donc ce que la couche IPsec doit faire des datagrammes IP alors que les SA explicitent le traitement à leur appliquer.

Lorsque les clés sont gérées dynamiquement par IKE, on définit manuellement des SPD symétriques sur chaque extrémité du VPN et on laisse les services IKE négocier et établir les SA.

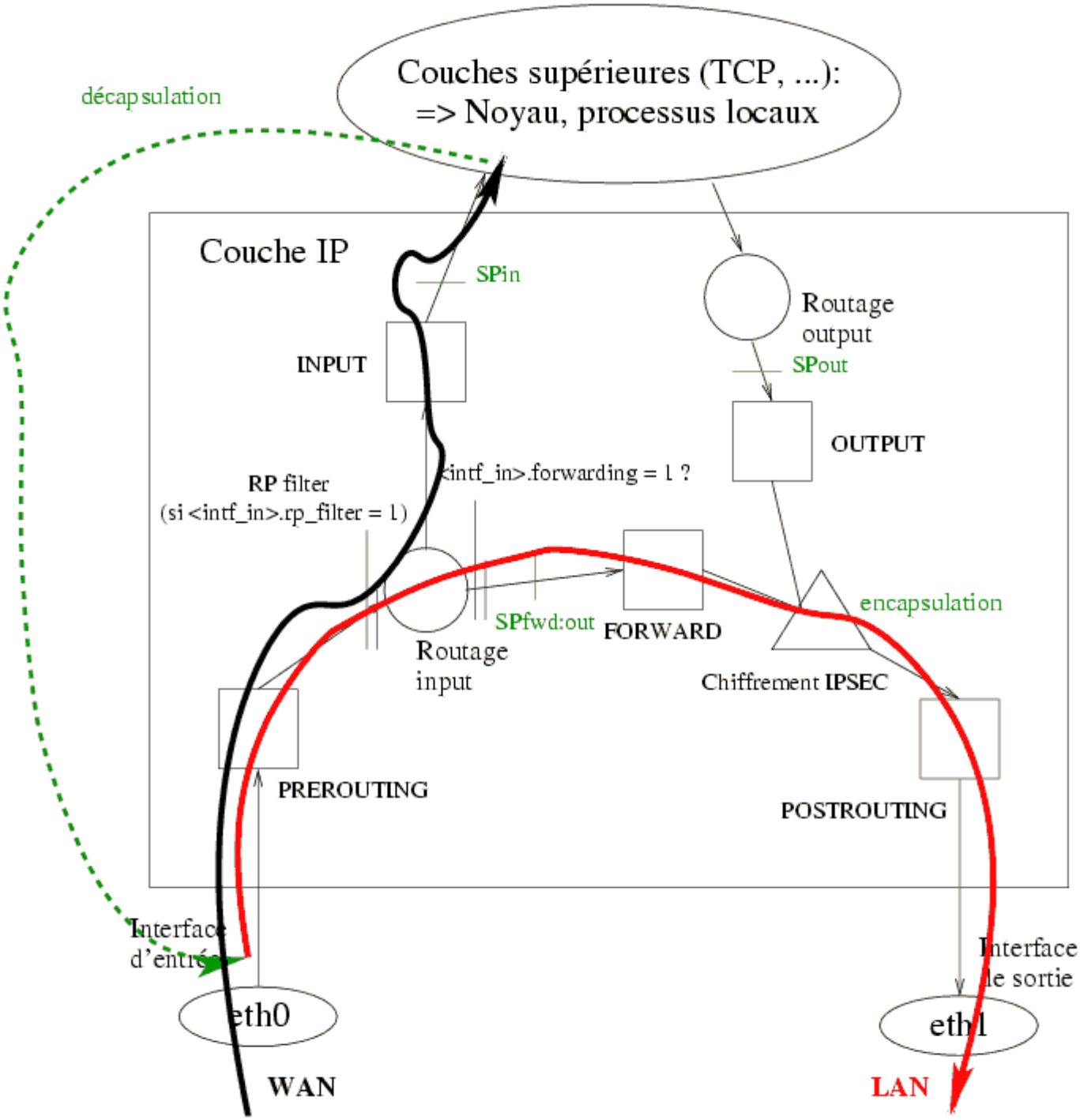
Cheminement d'un paquet à travers la pile IP

Les schémas qui suivent montrent le cheminement des paquets IP dans la pile IP d'un noyau Linux 2.6.

Ils illustrent respectivement le chiffrement et le déchiffrement d'un paquet au niveau **d'une passerelle VPN en mode tunnel.**
Il est important de bien visualiser ce cheminement pour éviter toute erreur lors de la configuration de la passerelle VPN, erreurs qui pourraient entraîner l'acceptation de flux non authentifiés, la fuite de flux clairs, etc.



Comme indiqué sur le schéma, l'encapsulation IPsec intervient juste avant la traversée de la chaîne POSTROUTING de Netfilter. Cette information permettra, par une règle de filtrage adaptée, de prévenir la sortie des flux clairs.



Les flux chiffrés sont d'abord reçus localement pour être déchiffrés par la couche ESP. Ils sont alors ré-injectés dans la pile IP comme s'ils étaient reçus en clair par leur interface d'entrée. Un marquage du paquet chiffré permettra, au niveau du filtrage, de distinguer entre un paquet arrivé en clair et un paquet qui a été authentifié et déchiffré.

La Pratique

La configuration IPsec du noyau Linux peut s'effectuer au moyen des outils natifs KAME ou via ISAKMPD (portage sous Linux des outils d'OpenBSD), qui exécutent des appels système pour

modifier les règles actives au niveau de la couche IPsec (à la manière d'iptables, qui permet de configurer Netfilter). Les outils utilisés dans le cadre du TP sont ceux de KAME.

Le réseau VPN déployé dans le cadre du TP met en oeuvre une gestion dynamique des clés (IKE) avec authentification par secret pré-partagé (pour des raisons de simplicité).

Les outils sont les suivants :

- setkey : cette commande sert à paramétrer les SA et les SP dans le noyau Linux. On peut l'utiliser en "ligne de commande" ou lui passer un fichier de configuration en argument.
- racoon : c'est le démon IKE qui établira les SA à la demande.

Récupérer l'archive sur le serveur FTP : **`ipsectools-0.3.1.tar.gz`**.

```
tar xvfz ipsectools-0.3.1.tar.gz

cd ipsectools-0.3.1

./configure --with-kernel-headers=/lib/modules/2.6.4/build/include

make

make install
```

Voici le fichier **`/etc/ipsec/ipsec.conf`** de configuration des SP sous Linux :

```
#!/usr/sbin/setkey -f

# Configuration pour FWI 192.168.T.3

# T  : numéro du groupe local

# T' : numéro du groupe distant



# Nettoyage des SA et de la SPD

flush;

spdflush;



# On ne configure aucune SA (utilisation de IKE)



# Création des politiques (simplistes)

spdadd 192.168.1T.0/24 192.168.1T'.0/24 any -P out ipsec
esp/tunnel/192.168.T.3-192.168.0.1T'/require;

spdadd 192.168.1T'.0/24 192.168.1T.0/24 any -P fwd ipsec
esp/tunnel/192.168.0.1T'-192.168.T.3/require;
```

```
spdadd 192.168.1T'.0/24 192.168.1T.0/24 any -P in ipsec
esp/tunnel/192.168.0.1T'-192.168.T.3/require;
```

Les politique IPsec pour les sens IN, OUT et FWD sont les mêmes : on veut imposer un chiffrement ESP en mode tunnel. On remarquera que la politique IN est normalement inutile puisqu'il s'agit d'une passerelle chiffrante et qu'aucun trafic ne lui est proprement destiné. Elle n'a été ajoutée que pour permettre à racoon d'en prendre connaissance (ce point sera vraisemblablement corrigé dans une version ultérieure).

On charge ces politiques par la commande **`setkey -f /etc/ipsec/ipsec.conf`**.

Il faut ensuite lancer le démon IKE. Voici le fichier de configuration **`/etc/ipsec/racoon.conf`** :

```
#!/usr/sbin/racoon -f

# Utilisable sur FWI 192.168.T.3


path pre_shared_key "/etc/ipsec/psk.txt";


# SA ISAKMP (main mode, avec secret prépartagé)

remote 192.168.0.1T' {

        exchange_mode main;

        proposal {

                encryption_algorithm 3des;

                hash_algorithm md5;

                authentication_method pre_shared_key;

                dh_group modp1024;

                }

}


# Proposition sortante (quick mode, en PFS)

sainfo address 192.168.1T.0/24 any address 192.168.1T'.0/24 any {

        pfs_group modp1024;

        encryption_algorithm 3des;

        authentication_algorithm hmac_md5;
```

```
        compression_algorithm deflate;

}
```

Le démon peut alors être démarré par la commande **`racoon -f /etc/ipsec/racoon.conf`**.

En fonctionnement, la SPD est consultée pour chaque paquet traversant la couche IP. Si celui-ci correspond à une entrée de la SPD (s'il faut le chiffrer), le noyau recherche la SA correspondante dans sa base. S'il ne trouve aucune SA convenable, le noyau contacte le démon racoon, qui se charge de négocier une SA avec la partie distante. L'encapsulation ESP peut alors être effectuée.

Le fichier de secret pré-partagé **`/etc/ipsec/psk.txt`** se présente ainsi :

```
# fichier contenant le secret.

# le format est le suivant :

# identité de la partie distante   |  secret pré-partagé

192.168.0.1T   mon_secret
```

Les règles de filtrage IP de la passerelle VPN (FWI) doivent être adaptées pour autoriser les flux clairs entre le LAN et le tunnel, ainsi que les flux IPsec et IKE sortants.

Voici un exemple de **Makefile** qui illustre la réalisation de règles visant à interdire les flux clairs sortants (absence de fuite d'information) et les flux entrants non issus d'un tunnel (refus des flux non authentifiés). Ces précautions limitent les risques de compromission du système en cas de mauvaise configuration ou en cours de rechargement des règles IPsec.

En ce qui concerne le FWE, le Makefile fourni dans le chapitre sur le filtrage inclut les règles de traitement du trafic IPsec.