Administration réseau sous linux (Debian et Ubuntu)

Rémy Malgouyres
LIMOS UMR 6158, IUT, département info
Université Clermont 1
B.P. 86
63172 AUBIERE cedex
http ://www.malgouyres.fr/

Certaines parties de ce document sont librement inspirées
du cours de Sébastien Salva
IUT, département info
Université Clermont 1
B.P. 86
63172 AUBIERE cedex
mais aussi de nombreux tutoriaux sur le web
qu'il est impossible de tous citer.
Merci à leurs auteurs.
J'ai aussi bénéficié de conversations instructives avec David Delon et Michael Witrant...

Une version PDF de ce document est téléchargeable sur mon site web, ansi que la version html.

Table des matières

Première partie

Configuration d'un réseau local (LAN)

Chapitre 1

Configuration d'une station

Un ordinateur communique avec les autres ordinateurs par des *interfaces réseaux*. En général, on a une interface réseau pour chaque carte réseaux (pour les ordinateurs qui ont plusieurs cartes réseaux). Une carte réseau (ethernet ou wifi) possède une adresse *MAC*, qui identifie la carte réseau sur le réseau. Pour faire fonctionner une carte réseau, il faut configurer l'interface qui li correspond.

1.1 Configurer les interfaces à la main avec `ifconfig`

1.1.1 Obtenir la configuration courante des interfaces

La commande `ifconfig` permet de connaître la configuration réseaux et de configurer le réseau à la main ou dans un script.

Voici le résultat de `ifconfig` sur une configuration simple, configuré pour éthernet. Dans cet exemple, la station a une interface `eth0`, qui a `192.168.0.2` comme adresse *IP* dans le sous réseau, ou réseau local `192.168.0` (appliquer le masque binaire `Mask` sur l'adresse *IP*). L'adresse *MAC* de la carte réseau est `00:B2:3A:24:F3:C4`.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:B2:3A:24:F3:C4
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2c0:9fff:fef9:95b0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:5
          collisions:0 txqueuelen:1000
          RX bytes:1520 (1.4 KiB)  TX bytes:2024 (1.9 KiB)
          Interrupt:10

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:92 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:0
RX bytes:6040 (5.8 KiB)  TX bytes:6040 (5.8 KiB)
```

Pour obtenir ce résultat, il faut que la machine soit connectée par un cable ethernet à un routeur (ou un modem-routeur) qui définisse le sous-réseau `192.168.0`. On configure alors la station avec `ifconfig` :

```
# ifconfig eth0 192.168.0.2
```

Pour déconfigurer l'interface `eth0`

```
# ifconfig eth0 down
```

(vérifier avec `ifconfig` : l'interface `eth0` n'apparaît plus).

1.1.2 Le fichier `resolv.conf` et la résolution des noms

Sur internet, les stations (appelée *hôtes* ont des adresses *IP*, qui permettent de les identifier, mais on peut aussi souvent désigner un hôte par un *nom d'hôte* ou nom de domaine (comme `example.com` ou `google.com`). Pour accéder à une machine à partir de son nom, notre station doit *résoudre l'hôte*, c'est à dire qu'elle doit trouver l'adresse *IP* de l'hôte à partir de son nom. Pour celà, notre station doit accéder à un serveur, appelé *serveur de noms* ou *DNS*. Ce server connaît les adresses *IP* correspondant à tous les noms d'hôte.

Le fichier `/etc/resolv.conf` contient les adresses *IP* d'un ou plusieurs serveurs de noms. Voici un exemple (avec des adresses bidon).

```
\# cat /etc/resolv.conf
search
nameserver 193.47.194.7
nameserver 193.47.194.9
```

Si l'on n'a pas accès à un serveur de noms, on ne peut accéder aux autres machines que par leur adresse *IP*, à l'exception des machines qui sont définies dans le fichier `/etc/hosts`.

```
# cat /etc/hosts
# adresse IP    Nom d'hôte
127.0.0.1       localhost
127.0.1.1       portable1

192.168.0.1    router
192.168.0.17  printer
208.77.188.166   example.com
...
```

1.2 Gérer la configuration dans le fichier `interfaces`

La configuration d'une interface avec `ifconfig` n'est pas enregistrée sur le disque, et en particulier, elle n'est pas conservée en cas de réinitialisation du système (*reboot*). Pour enregistrer la conficguration de manière permanente, if faut créer cette configuration dans un *fichier de configuration*.

Pour initialiser le réseau après configuration, il faut faire :

```
# /etc/init.d/networking start
```

Pour réinitialiser le réseau après un changement dans les fichiers de configuration, il faut faire :

```
# /etc/init.d/networking restart
```

1.2.1 Configuration des interfaces

La configuration des interfaces utilisée lors de l'initialisation du réseau est contenue dans le fichier `/etc/network/interfaces`.

1.2.1.a Pour ethernet static

```
# cat /etc/network/interfaces
# configuration de l'interface lo (obligatoire)
auto lo
iface lo inet loopback

# configuration de l'interface eth0
auto eth0
iface eth0 inet static
        address 192.168.0.2
        netmask 255.255.255.0
```

1.2.1.b Pour ethernet DHCP

Certains réseaux locaux sont configurés en *DHCP* (*pour Dynamic Host Configuration Protocol*). Dans un tel réseau, la station n'a pas besoin de connaître son adresse *IP* pour se connecter, mais l'addresse *IP* est fixée directement par le serveur *DHCP*. Dans ce cas, le fichier `interfaces` est nettement simplifié et ne dépend que de l'interface :

```
# cat /etc/network/interfaces
auto lo eth0
iface lo inet loopback

iface eth0 inet dhcp
```

1.3 Configuration *WIFI*

Le *WIFI* (protocole `802.11`) est une technologie de réseaux locaux sans fil. On accède au réseau via un serveur sans fil avec lequel on communique par des ondes électromagnétiques. Pour éviter que n'importe qui puisse se connecter au réseau sans fil, ou n'intercepte les communications, les communications sont cryptées. Il existe deux protocoles courant de cryptage : *WEP* et `WPA`. Voici le fichier `interfaces` configuré avec une clef de cryptage *WEP*. l'*ESSID* est le nom du réseau sans fil.

```
# cat /etc/network/interfaces
auto lo
```

```
iface lo inet loopback

# l'interface eth1 correspond ici à la carte wifi
auto eth1
iface eth1 inet dhcp
      wireless-essid mon_essid
      wireless-mode managed
      wireless-key AF32852BE7A39B522BG60C4353
```

Le *ESSID* et la clef *WEP* doivent correspondre et être correctement configurés sur le serveur sans fil.

Voici à quoi ressemble la configuration du réseau lors d'un connection sans fil :

```
# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:92 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6040 (5.8 KiB)  TX bytes:6040 (5.8 KiB)

eth1      Link encap:Ethernet  HWaddr 00:24:F2:C1:AB:23
          inet addr:192.168.0.3  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::213:d4ff:fef4:cd49/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:421 errors:0 dropped:0 overruns:0 frame:0
          TX packets:460 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4823848 (4.6 MiB)  TX bytes:79806 (77.9 KiB)
```

Pour obtenir les paramètres spécifiques au réseau *WIFI*, on utilise `iwconfig`

```
# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

eth1      IEEE 802.11g  ESSID:"my_essid"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 00:F0:C3:A4:C1:41
          Bit Rate:54 Mb/s   Tx-Power=20 dBm   Sensitivity=8/0
          Retry limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=0/100  Signal level=-23 dBm  Noise level=-86 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
```

```
          Tx excessive retries:5  Invalid misc:2   Missed beacon:98

sit0      no wireless extensions.
```

Chapitre 2

Serveur *DHCP*

2.1 Client *DHCP*

Le protocole *DHCP* (*pour Dynamic Host Configuration Protocol*). Dans un tel réseau, la station n'a pas besoin de connaître son adresse *IP* pour se connecter, mais l'addresse *IP* est fixée directement par le serveur *DHCP*. Dans ce cas, le fichier `interfaces` est nettement simplifié et ne dépend que de l'interface :

```
# cat /etc/network/interfaces
auto lo eth0
iface lo inet loopback

iface eth0 inet dhcp
```

Lorsque le client ainsi configuré démare le réseau, cela lance un processis `dhclient` qui cherche à contacter un serveur *DHCP*. Le serveur *DHCP* répond en allouant une adresse *IP* pour un certain temps.

2.2 Configuration d'un serveur *DHCP*

Pour redémarer les serveur *DHCP* après un changement de configuration, on dispose de la commande :

```
# /etc/init.d/dhcp3-server restart
```

Le fichier de configuration du serveur `dhcp3-server` est :

```
/etc/dhcp3/dhcpd.conf
```

- **`option domain-name`** `"monserveur.com"` : le ou les noms nom de domaine correspondant au réseau local.
- **`subnet`** Donne une idée au serveur *DHCP* de la topologie du réseau. Cette option ne change pas les accès ou les attributions d'adresses.
  Exemple :

```
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.2 192.168.0.20;
  option routers 192.168.0.1;
  default-lease-time 600;
  max-lease-time 7200;
}
```

Définit un sous-réseaux sur lequel les machines se connecteront par *DHCP*. Dans cet exemple, on précise les plages d'adresses allouées, ainsi que la durée pendant laquelle les *IP* sont réservées (*leased*) aux clients avant renouvellement.

- ```
  host guest {
    hardware ethernet 67:42:AB:E3:74:00;
    fixed-address 192.168.0.3;
  }
  ```

  Réserve une adresse *IP* fixe particulière un un certain client identifié par son adresse *MAC*.

On peut limiter les serveur `DHCP` à une (ou plusieurs) intefaces en éditant le fichier `/etc/default/dhcp3-se`
  ```

Chapitre 3

Partage de fichiers

3.1 Installation de *NFS*

Le protocole *NFS* (*Network File System*) permet de partager facilement des fichiers entre des machines *Unix*, et donc *Linux*. C'est un modèle client-serveur : une machine exporte, i.e. met à disposition, des répertoires de son système de fichier local sur le réseau. Suivant les droits d'accès, les autres stations du réseau peuvent monter ces répertoires, qui seront alors vus comme des répertoires locaux. Bien évidemment, un ordinateur peut être à la fois client et serveur *NFS*.

3.1.1 Installation coté serveur

On peut vérifier que les demons sont lancés, c'est à dire *nfs* avec la commande `ps`.

Vous pouvez lancer lancer les démons *NFS* manuellement avec :

```
\# /etc/init.d/nfs-kernel-server start
```

Pour redémarer le serveur *NFS* proprement :

```
\# /etc/init.d/nfs-kernel-server restart
```

On peut vérifier que les demons sont lancés :

```
/etc/init.d/nfs-kernel-server status
```

Attention : Certaines distributions de linux (comme la *Redhat*) utilisent le diffuseur de ports dynamiques (*portmapper*) : assurez-vous que ce service soit bien lancé au préalable :
`/etc/init.d/portmap status`
Sinon, démarrez-le : `/etc/init.d/portmap start`

3.1.1.a Configuration

Le fichier de configuration du serveur *NFS* est `/etc/exports`. On y indique la liste des répertoires qui peuvent être partagés (exportés). Une entrée dans ce répertoire se décompose de la manière suivante :
"répertoire local" "liste des machines autorisées à se connecter avec les options collées entre parenthèses"
par exemple :

```
/home    ollinux(rw)    station1(ro)\\
/projet    station1(rw) (ro)\\
/brouillon\\
```

L'ordinateur exporte son répertoire `/home` : la machine `ollinux` pourra le monter en lecture/écriture `(rw)`, la machine `station1` en lecture seule `(ro)`, les autres machine ne pourront pas se connecter. De même, `station1` pourra accéder en lecture/écriture au répertoire projet et toutes les autres stations en lecture seule. Enfin, tout le monde pourra accéder en lecture/écriture au répertoire brouillon (l'option `rw` est celle par défaut).

Notez bien que les droits en écriture via le réseau seront toujours inhibés par les droits sur le système de fichier. Prenons par exemple un fichier `test` appartenant à `root`, situé dans le répertoire projet et avec les droits 600 (lecture/écriture pour `root` uniquement , aucun droit pour les autres). Si l'utilisateur `toto` accède via la station `station1` au répertoire `/projet`, il ne pourra pas accéder au fichier `test`, bien qu'il ait les droits réseaux read-write.

Un fois le fichier `/etc/exports` correctement configuré, il suffit de relancer le service *NFS* par la commande suivante pour que les modifications soient prises en compte :

```
\textit{/etc/init.d/nfs-kernel-server reload}
```

3.1.2 Installation coté client

C'est relativement simple puisque le "système de fichier réseau" NFS est directement intégré au noyau. Il vous suffit de vérifier que ce dernier a été compilé avec la prise en charge de NFS. C'est le cas de toutes les distributions récentes.
Pour monter un système de fichier distant, utiliser la commande *mount* avec l'otion `nfs` :

```
$ mount -t nfs machine distante: répertoire\_partagé répertoire\_local -o options
```

Par exemple :

```
$ mount -t nfs 192.168.105.2:/armor/plages /mnt/cotes -o ro\\
```

montera le répertoire `/armor/plages`, exporté par la station `192.168.105.2`, dans le répertoire local `/mnt/cotes`, en lecture seule. À la place d'une adresse *IP*, vous pouvez toujours donner un nom de machine, comme par exemple `monhost`. Pour cela, il faut que le nom `monhost` soit résolu par un *DNS*. Pour un petit réseau, une simple entrée dans le répertoire `/etc/hosts` suffit, comme l'entrée suivante :
`192.168.105.2 monhost`
où `192.168.105.2` est l'adresse *IP* de `monhost`.

3.1.2.a Connection aux repertoires partages au démarrage.

Il est de plus possible de se connecter aux répertoires partagés au démarrage de la station. Le plus simple est de renseigner le fichier `/etc/fstab`, afin de faciliter le montage. La syntaxe est la suivante :

```
ordinateur-distant:répertoire-distant    répertoire-local nfs options 0 0
```

Pour reprendre l'exemple précédent, cela donnerait :

```
monhost:/armor/plages    /mnt/cotes nfs auto,rw,user,soft 0 0
```

Les options sont les mêmes que pour le système de fichier local (*ext2fs*) avec en plus, spécifiques à *NFS*.

3.2 Installation de *Samba*

Samba est un service permettant de partager des répertoires et imprimantes *Linux* entre des stations *linux* et des stations *Windows*.

3.2.1 Configuration et préparation du service *Samba*

Pour la configuration de ce service un seul fichier est à modifier `smb.conf` qui se trouve généralement dans le répertoire `/etc/samba` (ou `/etc` selon la distribution). Nou étudions ici le simple partage de répertoires avec autentification par login et mot de passe.

Dans le fichier de configuration `smb.conf`, la configuration générale est la suivante :

- **`workgroup =`** nom du groupe de travail windows (qui est en général Workgroup par défaut sous windows)
- **`server string =`** nom_du_serveur (c'est le nom qui va être vu dans voisinage réseaux)
- **`security = user`** pour que seuls les titulaires d'un compte *unix* sur le serveur puissent utiliser *samba*.
- **`encrypt passwords = true`** (pour activer le cryptage de password. Attention au `s` de passwords)
- **`log file =`** `/var/log/samba/log.%m` (chemin du fichier log)
- **`max log size =`** `50` (taille du fichier log)
- **`socket options =`** `TCP_NODELAY` (permet d'avoir de meilleur performance réseaux
- **`dns proxy =`** `No` (permet d'activer la fonction de résolution de noms NetBios)

Quelques options peuvent de plus être ajoutées :

- **printcap name** = /etc/printcap (chemin du fichier qui contient la liste des imprimantes)
- **load printers** = yes (va avec printcap et permet de charger toutes les imprimantes installé sur le system, ça évite de les configurer une à une dans le smb.conf)

D'autres options sont possibles pour par exemple configurer samba pour qu'il utilise un serveur windows Nt comme domaine primaire, ou pour qu'il résolve les noms d'hôtes, etc... (voir les `howto` sur le *web*).

3.2.2 Configuration du partage des répertoires

Maintenant notre serveur Samba est prêt pour pouvoir partager des répertoires. Il nous faut les configurer. La configuration des répertoires se fait comme suit :

- **Options commune à tous les répertoires (`[homes]`)**

```
writable = yes # permet l'accès en écriture
create mask = 0775 # permet l'accès en écriture
directory mask = 0775 # permet l'accès en écriture
```

- `[nom_du_répertoire]` (le nom qui sera visible dans voisinage réseaux)
- **`comment =`** `commentaire`
- **`path =`** `/mon/repertoire/` (le path du répertoire sur la machine linux)
- **`writable =`** `yes` (pour activer la lecture seule)
- **`public =`** `no` rend le répertoire privé (identification requise pour l'accès au répertoire)
- **`valid users =`** `liste d'utilisateurs` autorise seulement ces utilisateurs à accéder au répertoire. A utiliser avec public=no
- **`hosts allow =`** `192.168.0` restreint l'accès à partir des sous-réseaux ou d'une adresse *IP*.
- **`browseable =`** `no` (n'autorise pas le parcours du répertoire

Voici un exemple de fichier de configuration. Notons que le fichier `smb.conf` contient déjà des options par défaut lors de l'installation et qu'il faut seulement modifier les options si besoin créer les répertoires partagés.

```
   workgroup = MON_WORKGROUP

   server string = zebigboss server

   dns proxy = no
   security = user
   encrypt passwords = true

   socket options = TCP_NODELAY


[homes]
   comment = Home Directories
   browseable = no
```

```
   writable = yes
   create mask = 0775
   directory mask = 0775

   valid users = toto, webmaster

# déclaration d'un répertoire partagé "partage"
# ouvert en écriture dans /home/partage
[partage]
  comment = Repertoire partagé
  browseable = no
  writeable = nes
  public = no
  path = /home/partage
```

Avec l'option `public = no` ou les options `valid users =`, les utilisateurs doivent s'authentifier lors de la connexion aux ressources partagées. Il faut alors créer des comptes *samba* pour tous les utilisateur concernés (ces comptes *samba* peuvent éventuellement être syncronisés avec les comptes *unix*.

Pour créer un compte *samba* pour l'utilisateur `toto` :

```
\# smbpasswd -a toto
```

Cette commande demande le mot de passe samba de l'utilisateur.

3.2.3 Prise en compte des modifications et lancement du service

Pour lancer *samba* :

```
/etc/init.d/samba start
```

Pour arrêter *samba* :

```
/etc/init.d/samba stop
```

Pour redémarer proprement *samba* après un changement de configuration dans `smb.conf` :

```
/etc/init.d/samba restart
```

3.2.4 Accès aux répertoires partagés

3.2.4.a A partir de *linux*

Vous pouvez accéder aux répertoires partagés sous un autre station linux avec la commande :

```
smbclient //nom_du_serveur/nom_du_répertoire -U nom_utilisateur -I IP_du_server
```

Dans ce cas, smbclient se comporte comme un client *ftp*, avec les commandes `get`, `put`, etc... pour transférer les fichiers (tapez `help` pour la liste des commandes et `help cmd` pour l'aide sur la commande `cmd`).

3.2.4.b A partir de *Windows*

Dans l'interface de *Windows XP*, accédez au voisinage réseau par :

- Favoris réseaux ;
- Voir les ordinateurs du groupe de travail ;
- Réseau *Microsoft Windows*

Seuls les répertoires avec l'option `browsable = yes` apparaîssent dans l'interface de l'expolrateur *Windows*.

Deuxième partie

Applications *WEB*, *FTP* et *Mail*

Chapitre 4

Initiation à la programmation *WEB*

4.1 Principales balises *HTML*

Le langage *HTML* (ou *Hyper Tex Modeling Language*) permet d'écrire en texte *ASCII* du texte qui sera interprété par les navigateurs *Web* pour afficher des doncuments mis en forme.

La mise en forme, ainsi que les liens hypertexte, etc... sont indiqués dans le source *HTML* par des *balises*. Voici un certain nombre de balises *HTML*.

Voici un exemple de fichier source `HTML`, dont le rendu sous les navigateur *Iceweasel* est montré sur la figure 4.1.

FIGURE 4.1: Exemple de page *HTML*

```
<html>
<head>
<title>Ma première page HTML</title>
</head>
```

```
<body>

<center><h1>Ceci est un gros titre</h1></center>

<p>
On met le texte dans un paragraphe
<br>
On peut aller à la ligne avec une balise BR
</p>

<p>
Quand on crée un nouveau paragraphe, on saute une ligne.
</p>

<p>
On peut mettre un lien : <a href="http://google.fr">Cliquez ici</a>
pour aller sur google.
</p>
</body>
</html>
```

Voici quelques balises *HTML* indispensables :

- `<HTML> </HTML>` Balises de début et de fin de fichier *HTML*.
- `<HEAD> </HEAD>` Balises de début et de fin de l'entête du fichier (l'en-tête du fichier peut contenir le titre, la couleur du texte, la couleur du fond,...). `<TITLE> </TITLE>` Pour ajouter un titre ; se place généralement dans l'entête.
- `<BODY> </BODY>` Le corps de la page html (après l'entête) ; contient le corps du document proprement dit.
- `<br>` Passage à la ligne de ligne.
- `<CENTER> </CENTER>` Permet de centrer du texte.
- `<Hx> </Hx>` Permet de placer un titre x compris entre 1 et 6
- `<img scr="/chemin/vers/l'image">` Insérer une image
- `<A HREF="http://adresse_url"> </a>` Place un hyperlien vers un document ou un programme *CGI* (voir plus loin).

Il existe de nombreuses autres balises, que l'on peut trouver dans un cours ou tutorial d'*HTML*.

4.2 Réaliser un script *CGI*

Un script *CGI* est un programme exécutable sue le serveur, qui peut être réalisé dans n'importe quel langage (*C*, *C++*, *shell bash*,...). Les scripts `CGI` doivent être placés par le webmaster dans

certains répertoires particuliers (généralement appelés `cgi-bin`). La sortie standard du script génère du code (par exemple du code *HTML*) interprétable par un navigateur. Le navigateur affiche la page générée.

L'appel à un script `essai.cgi` dans le répertoire `cgi-bin` à la racine du serveur sera effectuée par l'ouverture de l'*URL* :
`http://serveur/cgi-bin/essai.cgi`
dans le navigateur.
Voici un exemple de script `CGI` écrit en *bash*. La première ligne de la sortie standard du script, contenant le `Content-type`, est obligatoire et décrit le langage du code généré par le script.

```
#!/bin/bash
echo -e "Content-type: text/html\n\n"
echo "<html>"
echo "<body>"
echo -n "Bonjour, ceci est le script \$(basename \$0)"
echo "du r&eacute;pertoire \$(pwd)"
echo "<BR>"
echo "Nous sommes le \$(date +%d/%m/%Y) et il est \$(date +%Hh%Mmn)".
echo "</body>"
echo "</html>"
```

4.3 Les formulaires

Un formulaire est un élément d'une page HTML permettant à l'inrenaute (le client) de saisir des données qui sont transmises à un script *CGI* sur un serveur `http`. Le script construit dynamiquement une page *HTML* puis l'envoie au client.

La syntaxe *HTML* d'un formulaire est la suivante :

```
<FORM NAME="nom\_du\_formulaire"
ACTION="url du programme que vous appelez"
METHOD= POST {\em ou} GET>
Corps du formulaire
</FORM>
```

Deux méthodes d'appel de programme sont proposées : les méthodes *GET* ou *POST*.

- **Méthode Get :** Par défaut, la méthode *GET* est utilisée. Les arguments que vous pouvez donner au programme appelé (voir `INPUT`) passent dans la variable d'environnement `QUERY_STRING` et sont visibles depuis le navigateur. **Il faut le moins possible utiliser cette méthode**, notamment pour des raisons de sécurité.

- **Méthode Post :** Dans le cas d'une méthode *POST*, les arguments sont émis vers l'entrée standard `stdin` du programme appelé. Les données ne sont pas visibles par le navigateur.

Le corps du formulaire peut être composé de texte et (surtout) de d'entrées (`INPUT` ou `SELECT`) qui permettent à l'utilisateur d'entrer des valeurs qui seront émises vers le programme cgi.

4.4 INPUT et SELECT

Les balises `INPUT` et `SELECT` permettent de faire saisir par le client les données qui peuvent être transmises à un script *CGI*.

4.4.1 INPUT

La balise `INPUT` crée des des champs dans une page *HTML* permettant de recueillir soit du texte, soit un choix parmis une liste, soit un booléen. Voici la syntaxe d'`INPUT` :

```
<INPUT TYPE="text" {\em ou} "submit" {\em ou} checkbox {\em ou} radio...
       NAME=nom\_de\_l'entrée
       SIZE=taille du champs
       VALUE="valeur par defaut du champ ou étiquette du bouton submit"
>
```

- Le **type "text"** indique qu'il s'agit d'une input permettant de recueillir des données texte.
- Le **type checkbox** crée une case à cocher.
- Le **type "submit"** permet la création d'un bouton. Lorsque l'on clique dessus et que l'on a crée un formulaire, le programme cgi est appelé.
- Dans tout les cas, l'attribut **NAME** indique le nom de l'entrée. Il est inutile avec le type submit mais dans les autres cas il permet au script *CGI* de s'y retrouver entre les différents champs du formulaire.
- **Size** indique la taille d'un champ de type texte dans la page *HTML* (nombre de caractères).
- **VALUE** est la valeur par défaut du champ ou l'étiquette du bouton ou de la case.

4.4.2 SELECT

La balise `SELECT` crée une liste déroulante où l'utilisateur n'a plus qu'à sélectionner sur l'un des choix proposés. Voici sa syntaxe :

```
<SELECT NAME="nom\_de\_l'entrée">
   <OPTION> un des choix que vous voulez mettre
   <OPTION> un autre choix
</SELECT>
```

On peut placer autant de choix que l'on souhaite.

4.5 Un exemple

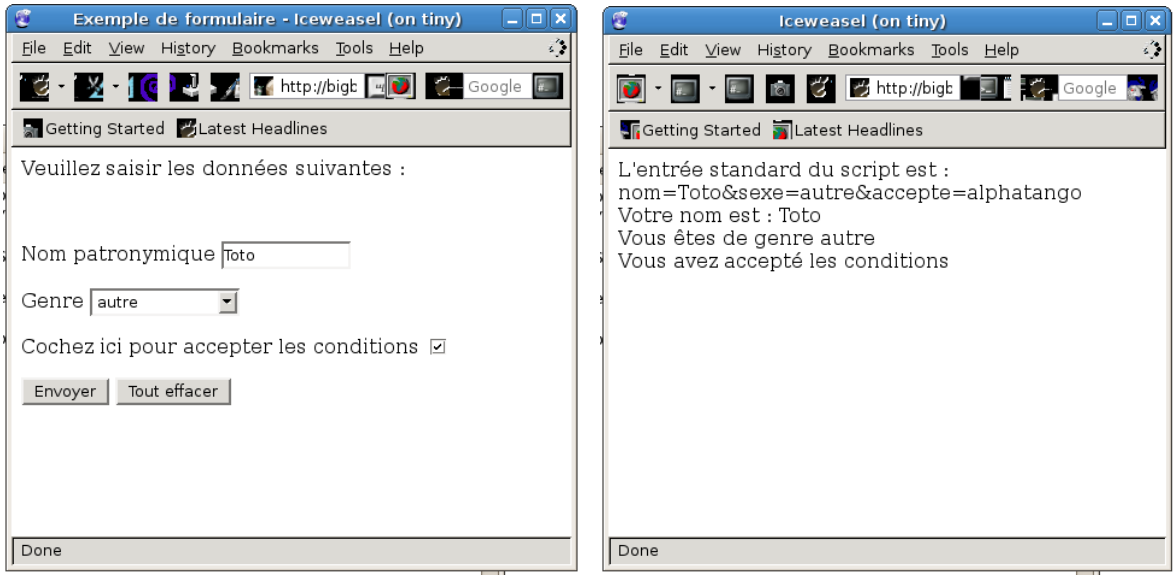
Voici (voir figure 4.2) un exemple de formulaire *HTML* avec les script *CGI* écrit en *bash* qui traite les données. Ici, le chemin au script `my_script.cgi` est donné en relatif.

```
<html>
  <head>
    <title>Exemple de formulaire</title>
  </head>
<body>
  <p>
    Veuillez saisir les donn&eacute;es suivantes :
  </p>
  <br>
  <form name="formulaire exemple"
        action=/cgi-bin/my_script.cgi method="post">
    <p>
      Nom patronymique
      <input type="text" name=nom size=12 value="">
    </p>
    <p>
      Genre
      <select name=sexe value="Choisissez">
         <option>Choisissez
         <option>Homme
         <option>femme
         <option>autre
      </select>
    </p>
    <p>
      Cochez ici pour accepter les conditions
      <input type="checkbox" name=accepte value="true" unchecked>
    </p>
    <p>
      <input type="submit" value="Envoyer">
      <input type="reset" value="Tout effacer">
    </p>
  </form>
</body>
</html>
```

Voici le script `my_script.cgi`

```
#!/bin/bash
echo -e "Content-type: text/html\n\n"
read ENTREE

echo "L'enr&eacute;e standard du script est :"
```



(a) Le formulaire *HTML* (b) Le résultat du script

FIGURE 4.2: Exemple de formulaire *HTML*

```
echo "<br>"
echo $ENTREE
echo "<br>"

echo "Votre nom est : "
echo $ENTREE | cut -d'&' -f1 |cut -d'=' -f 2
echo "<BR>"
echo "Vous &ecirc;tes de sexe "
echo $ENTREE | cut -d'&' -f2 |cut -d'=' -f 2
echo "<BR>"
CHAMP3=$(echo $ENTREE | cut -d'&' -f3 |grep true)

if [ -n "$CHAMP3" ]
then
        echo "Vous avez accept&eacute; les conditions"
else
        echo "Vous avez refus&eacute; les conditions"
fi
```

Une alternative à la `checkbox` est le type `radio`, qui crée deux choix qui s'excluent mutuellement.

```
<form>
...
  <input type="radio" name=accepte value="true"> J'accepte les conditions
  <br>
  <input type="radio" name=accepte value="false"> Je refuse les conditions
...
```

```
</form>
```

4.6 Notions de *PHP*

4.6.1 Mon premier script *PHP*

Le *PHP* est un langage de scripts qui s'insère dans du code *HTML*, et qui, en s'exécutant coté serveur, permet de générer du code *HTML* qui est affiché coté client. La différence avec les script *CGI* est dans la manière de passer les paramètres d'une page à l'autre. De plus, le *PHP* est un langage spécialement conçu pour la programmation *WEB*. Il permet de créer des sites complètement dynamiques, c'est à dire que les contenus des pages dépendent entièrement des actions ou de l'identité du client.

Le code *PHP* est inséré dans une balise `<?php ?>`. Les instructions se terminent pas des point-virgules. Voici un exmple du script *PHP* le plus simple, qui affiche simplement le mot "bonjour" :

```
<html>
<head>
  <title>Mon premier programme PHP</title>
</head>
<body>
   <?php echo '<p>Bonjour !!</p>'; ?>
</body>
</html>
```

On voit que le script génère du code *HTML* par la commande `echo`.

L'exemple précédent peut facilement être réalisé en *HTML*. Ca n'est pas le cas de l'exemple suivant, qui affiche la date :

```
<html>
<head>
  <title>Afficher la date en PHP</title>
</head>
<body>
  <?php
    echo date('d/m/Y- H:i');
  ?>
</body>
</html>
```

La fonction `mail` permet à un script *PHP* d'envoyer un mail. Cette fonction prend en paramètre l'adresse du destinataire, le sujet, et le corps du message.

```
<?php
  mail('webmaster@localhost',
       'Erreur 202',
       'Il s'est produit une erreur 202 sur le serveur');
?>
```

Voici maintenant un exemple d'utilisation d'un tableau et d'une boucle en *PHP* :

```
<html>
<head>
  <title>Exemple de boucle et de tableau</title>
</head>
<body>
  <?php
    \$prenom = array('Olivier','Fran&ccedil;ois','S&eacute;bastien','Jean-Louis');
    echo 'Voici les quatre pr&eacute;noms du tableau : <BR>';
    echo \$prenom[0].', '.\$prenom[1].', '.\$prenom[2].' et '.\$prenom[3].'<br>';

    for(\$i=0;\$i<sizeof(\$prenom);\$i++)
     {
       echo \$prenom[\$i].'<br>';
     }
  ?>
</body>
</html>
```

4.6.2 Traitement d'un formulaire

Voici le traîtement d'un formulaire par un script *PHP*. Le formulaire est le même que pour l'exemple de script *CGI* ci-dessus excepté que le script appelé dans le champ `action` du formulaire est un script *PHP*. Voici le fichier `formulaire.html` :

```
<html>
  <head>
    <title>Exemple de formulaire</title>
  </head>
<body>
  <p>
    Veuillez saisir les donn&eacute;es suivantes :
  </p>
  <br>
  <form name="formulaire exemple"
        action=./my_script.php method="post">
    <p>
      Nom patronymique
      <input type="text" name=nom size=12 value="">
    </p>
    <p>
      Genre
      <select name=sexe value="Choisissez">
         <option>Choisissez
         <option>Homme
         <option>femme
         <option>autre
```

```
      </select>
    </p>
    <p>
      Cochez ici pour accepter les conditions
      <input type="checkbox" name=accepte value="true" unchecked>
    </p>
    <p>
      <input type="submit" value="Envoyer">
      <input type="reset" value="Tout effacer">
    </p>
  </form>
</body>
</html>
```

Voici le fichier my_script.php :

```
<html>
<head>
   <title>Traitement d'un formulaire en PHP</title>
</head>
<body>
  <?php
    if (\$_POST['sexe'] == 'Choisissez')
      {
        echo 'Erreur, vous devez obligatoirement faire un choix ';
        echo '<a href="./formulaire.html">Cliquez ici</a> pour vous ressaisir.';
      }else
      {
        echo 'Votre nom est ';
        echo htmlspecialchars(\$_POST['nom']);
        echo '<br>Vous &ecirc;tes de genre ';
        echo \$_POST['sexe'];
        if (\$_POST['accepte'] == 'true')
          echo '<br>Vous avez accept&eacute; les conditions';
        else
          echo '<br>Vous avez refus&eacute; les conditions';
      }
?>
</body>
</html>
```

Chapitre 5

Configuration d'Apache

Apache est un serveur http libre, c'est un des serveurs http les plus utilisé sur Internet avec plus de 60% des sites d'Internet (contre environ 20% pour IIS).

En résumé, un serveur http est un serveur hébergeant un ou plusieurs sites Web c'est à dire des pages html ou des programmes générant des pages html (programmes cgi) qui sont accessibles par des navigateurs de type mozilla ou autre. Le protocole, permettant l'échange de pages html est le protocole http, d'ou le nom de serveur http. Ce protocole utilise généralement le port 80.

On trouvera une documentation complète sur apache (en anglais) sur le site suivant : `http://httpd.apache.org/docs/`.
Premièrement, sous Debian, il faut savoir lancer le démon apache :

```
# /etc/init.d/apache2 start.
```

Pour vérifier que le démon apache tourne :

```
# ps -ef|grep apache
```

Pour relancer apache après un changement de configuration, tapez :

```
# /etc/init.d/apache2 start.
```

5.1 Configuration de base

La configuration globale d'apache s'effectue par modification du fichier de configuration */etc/apache2/apach*
Apache pouvant gérer plusieurs serveurs, on trouve des compléments pour la configuration de chaque serveur dans `/etc/apache2/site-enabled`. Les fichiers de `/etc/apache2/site-enabled` sont inclus dans *apache2.conf* par un *Include* et suivent la même syntaxe. Les fichiers dans `/etc/apache2/site-enabled` correspondent aux serveurs activés et sont en fait un lien symbolique vers un fichier dans `/etc/apache2/site-available`, qui contient la liste de tous les serveurs disponibles.
Par défaut, il n'y a qu'un seul serveur activé : le serveur par défaut.

Ce qui suit décrit les principaux paramètres, et leur valeur attribuée par défaut à l'installation.

5.1.1 Paramètres généraux

Les paramètres qui sont (en général) **valable pour tous les serveurs** se trouve dans `apache2.conf` :

`User` `www-data`
fixe l'utilisateur (ici un groupe) qui peut posséder des scripts et données sensibles.
`Group` `www-data`
fixe le groupe qui peut posséder des scripts et données sensibles. Si on veut utiliser un utilisateur à posséder (par exemple) des scripts *cgi*, on doit ajouter cet utilisateur à groupe `www-data` par `adduser`.

`AccessFileName` `.htaccess`
Cette clause fixe le nom du fichier (par défaut .htaccess) à trouver dans un répertoire pour que l'accès de ce répertoire soit protégé, en imposant à l'utilisateur une authentification par nom et mot de passe. Ces comptes sont spécifiques à Apache et n'interfèrent pas avec les comptes Linux.

`Port` `80`
Apache écoute sur le port `tcp` usuel

`ServerRoot` `/etc/apache2`
Il s'agit du répertoire où le serveur trouvera son répertoire de configuration. On trouve dans /etc/apache, un lien vers /var/log/httpd/access_log, le fichier-journal des accès aux ressources, réussis ou non (le consulter)

`PidFile` `/var/run/httpd.pid`
C'est le fichier où le serveur en exécution stocke son premier numéro de processus (PID), ce qui peut être utile à d'autres processus.

`ErrorLog` `/var/log/apache2/error.log`
C'est le fichier qui contient l'historique des erreurs qui se sont produites (exemple : script `cgi` qui n'a pas marché...).

5.1.2 Paramètres spécifiques à chaque serveur

Les paramètres (en général) ă**spécifiques à chaque serveur** (qui se trouvent dans *sites-enabled* sont (liste non exhaustive) :

`DocumentRoot` `/var/www/`
fixe la racine du serveur Web, c'est-à-dire le répertoire de base où sont cherchées par défaut les pages html, lorsque l'URL ne comporte pas de chemin de répertoire

`DirectoryIndex` `index.html index.php index.htm...`
Il est courant d'omettre le nom du fichier de la page d'accueil d'un site ou de l'un de ses sous-répertoires. Pour ne pas retourner systématiquement une erreur 404 signalant une adresse erronnée, le serveur possède une liste standard de noms de fichiers qu'il s'efforce de trouver dans le répertoire. Cette liste ordonnée est indiquée par la clause DirectoryIndex

`ServerAdmin` `webmaster@localhost`
S'il a un problème, le serveur écrit un message à cette adresse

CustomLog /var/log/apache2/access.log combined

`CustomLog /var/log/apache2/access.log combined`
Définit le fichier qui contient l'historique des connections, des clients, des dates, de l'origine (*referer site*) de la connection, ainsi que le format pour mémoriser ces informations (ici le format combined).

5.2 Contrôle des accès à un répertoire

Chaque répertoire auquel Apache accède peut être configuré, et `root` peut permettre certaines fonctionalités d'apache pour ces répertoires, et en interdire d'autres. Cela permet, en fonction des besoin et de la confiance accordée à chaque webmaster, de gérer les problèmes de sécurité. En général, `root` cherche à donner tout juste les permissions qui sont requises en fonction des besoins. Le paramétrage d'un répertoire se précise dans une balise notée :
`<Directory /chemin/vers/le/repertoire/> </Directory>`

5.2.1 Exemple.

```
NameVirtualHost *
<VirtualHost *>
        DocumentRoot /home/monRepertoire/  # racine du site
        <Directory />   # droits du répertoire racine
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /home/monRepertoire/> # droits sur l'ensemble du site
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None # interdit les .htaccess
                Order allow,deny  # donnne l'ordre des permissions
                allow from all    # autorise tous les clients
                # avec la directive suivante, il faut mettre la
                # page d'acueil dans /home/monRepertoire/apache2-default/
                # Commentez si vous voulez mettre les fichiers à la
                # racine de /home/monRepertoire/
                RedirectMatch ^/\$ /apache2-default/
        </Directory>

        etc...
</VirtualHost>
```

5.2.2 Principales options

Les principales options d'un répertoire peuvent être les suivantes :

1. **None** : Désactive toutes les options.

2. **All** : Active toutes les options SAUF Multiviews.

3. **Indexes** : Permet aux utilisateurs d'avoir des index généré par le serveur. C'est à dire si l'index du répertoire (index.html ou index.php par exemple) est manquant, cela autorise le serveur a lister le contenu du répertoire (dangereux suivant les fichiers contenu dans ce répertoire).

4. **FollowSymLinks** : Autorise a suivre les liens symboliques.

5. **ExecCGI** : Autorise à exécuter des scripts CGI dans ce répertoire.

6. **Includes** : Autorise des fichiers include coté serveur *SSI*.

7. **IncludesNOEXEC** : Permet mais les includes mais empêche la commande EXEC (qui permet d'executer du code).

8. **Multiviews** : Autorise les vue multiples suivant un contexte. Par exemple permet d'afficher les pages dans une langue différente suivant la configuration du client.

9. **SymLinksIfOwnerMatch** : Autorise a suivre les liens seulement si l'user ID du fichier (ou répertoire) sur lequel le lien pointe est le même que celui du lien.

5.2.3 Donner les droits

Avec `Order allow,deny`, on peut permettre un accès à tous sauf quelques-uns. Par exemple,

```
Order allow,deny
allow from all    # autorise tous les clients
deny from 192.168.0.67 # interdit l'accès par une IP
```

permet à tous d'accéder sauf l'hôte `192.168.0.67`.

Avec `Order deny, allow`, on peut permettre l'accès seulement par un sous-réseau. Par exemple,

```
Order deny,allow
Deny from all
Allow from 192.168.0.0/255.255.255.0
Allow from .mydomain.com
```

permet l'accès seulement à partir du sous-réseau `192.168.0.0/24` et du domaine `mydomain.com`.

5.2.4 Directive AllowOverride

La directive `AllowOverride` permet à l'administrateur d'autoriser le webmaster à redéfinir par lui-même certains droits ou certaines options spécifiquement dans certains répertoires. Pour cela, le webmaster crée dans un répertoire un fichier `.htaccess` dans lequel il définit les options et les droits qu'il souhaite.

Par exemple, si `root` a mis dans les permissions d'un répertoire

```
AllowOverride Options Limit
```

le webmaster peut mettre les droits suivants dans un fichier `.htaccess` d'un répertoire contenant des fichiers de l'intranet de son entreprise :

```
Options ExecCGI
Order deny,allow
Deny from all
Allow from 192.168.0.0/255.255.255.0
```

Les types de directives que l'on peut mettre après `AllowOverride` son les suivants

1. **None** : N'autorise aucun contrôle par le webmaster au niveau du `.htaccess`. Apache ne lis pas le fichier `.htaccess` et laisse les permissions "Linux" de ce répertoire et les droits donnés par `root` dans la balise `<Directory>` dans la configuration d'apache.
2. **All** : toutes les permissions et options peuvent être gérés dans par le webmaster dans le `.htaccess` d'un répertoire.
3. **Limit** : Active la directive d'autorisation `order, allow, deny` dans le `.htaccess`.
4. **Options** : Active la directive `Options` dans le `.htaccess`.
5. **AuthConfig** : permet au webmaster de configurer dans le `.htaccess` les directives d'autentification pour les sites sécurisés (AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile AuthName, AuthType, AuthUserFile, Require).
6. **FileInfo** : Active les directives d'autorisations AddEncoding, AddLanguage, AddType, DefaultType, ErrorDocument, LanguagePriority.
7. **Indexes** : permet de définir dans `.htaccess` des directives comme `DirectoryIndex`.

5.3 Configurer l'accès aux scripts *CGI*

Un script *CGI* est un programme qui peut être exécuté sur le serveur. Il peut être réalisé dans n'importe que langage (script shell, *C*, perl,...). Il doit vérifier certaines normes sur les entrées-sorties. Lorqu'un client web appele le script sur son navigateur, le script est exécuté coté serveur, et son éventuelle sortie est affichée.

Les *CGI* ont normalement été intégrés au serveur Apache sous forme d'un module chargeable, le fichier `mod_cgi.so`, situé comme tous les autres modules d'Apache dans `/usr/lib/apache2/modu`

La liste des modules activés se trouve dans le répertoire
`/etc/apache2/mods-enabled/`
Vérifiez que le module `cgi` s'y trouve bien. S'il n'y est pas, créez des liens symbolique vers les fichiers correspondant dans `/etc/apache2/mods-available` :

```
# cd /etc/apache2/mods-enabled/
# ln -s ../mods-available/cgi.* .
```

ou plus simplement :

```
# a2enmod cgi
```

Pour qu'Apache prenne en charge un script, il est nécessaire d'effectuer un minimum de paramétrage. Il faut ajouter la directive `ScriptAlias` qui précise le nom du seul répertoire autorisé à contenir des scripts :

```
ScriptAlias /cgi-bin /monRepertoire/cgi-bin/
```

ainsi que la configuration du répertoire contenant les scripts :

```
<Directory /monRepertoire/cgi-bin>
      AllowOverride None
      Options ExecCGI
</Directory>
```

5.4 Le module *PHP*

PHP a normalement été intégré au serveur Apache sous forme d'un module chargeable, le fichier `libphp5.so`, situé comme tous les autres modules d'Apache dans `/usr/lib/apache2/modules/`.

La liste des modules activés se trouve dans le répertoire
`/etc/apache2/mods-enabled/`
Vérifiez que le module `php5` s'y trouve bien. S'il n'y est pas, créez des liens symbolique vers les fichiers correspondant dans `/etc/apache2/mods-available` :

```
# cd /etc/apache2/mods-enabled/
# ln -s ../mods-available/php5.* .
```

ou plus simplement :

```
# a2enmod php5
```

On peut voir la configuration du serveur pour le `php` :

```
# cat /etc/apache2/mods-enabled/php5.conf
<IfModule mod_php5.c>
  AddType application/x-httpd-php .php .phtml .php3
  AddType application/x-httpd-php-source .phps
</IfModule>
```

5.5 Gérer les pages *WEB* personnelles

L'administrateur d'apache peut permette à tous (ou certains) utilisateurs de publier leurs propres contenus *WEB*. Le moyen le plus simple pour celà est d'utiliser le module `userdir`. Cela permet de donner les droits sur un répertoire pour chaque utilisateur, classiquement le répertoire `/public_html` pour que ce répertoire soit accessible sur le web.

La liste des modules activés se trouve dans le répertoire
`/etc/apache2/mods-enabled/`
Vérifiez que le module `userdir` s'y trouve bien. S'il n'y est pas, créez des liens symbolique vers les fichiers correspondant dans `/etc/apache2/mods-available` :

```
# cd /etc/apache2/mods-enabled/
# ln -s ../mods-available/userdir.* .
```

ou plus simplement :

```
# a2enmod userdir
```

On peut voir la configuration du serveur pour les pages perso :

```
# cat /etc/apache2/mods-enabled/userdir.conf
<IfModule mod_userdir.c>
        UserDir public_html           # pour tous les utilisateurs
        UserDir disabled root badguy  # désactivé pour certains

        # règles d'accès et droits pour les pages perso
        <Directory /home/*/public_html>
                AllowOverride FileInfo AuthConfig Limit
                Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
        </Directory>
</IfModule>
```

5.6 Virtual hosts

Une machine peut en général avoir plusiers noms d'hôte. One peut en déclarer plusieurs pour la même adresse *IP* dans les *DNS*, et une machine peut aussi avoir plusieurs adresses *IP* si elle a plusieurs interfaces réseaux.

Si une machine a plusieurs noms d'hôte, on peut alors mettre plusieurs sites *HTTP* sur le même serveur.

Pour celà, on crée plusiers fichiers de configuration différents dans le répertoire `/etc/apache2/sites-enabled`
Dans la déclaration du virtual host, on peut mettre :

```
NameVirtualHost *
<VirtualHost *>
   ServerName mon_nom_d_hote
...
```

suivi de la déclaration du répertoire racine du site. En général, on met bien évidement plusieurs répertoires racine différents pour les différents hôtes, ce qui permet d'avoir plusieurs sites *WEB*.

5.7 Gestion des ressources du serveur

5.8 Statistiques *WEB* avec *awstats*

L'outil de statistiques *WEB* *awstats* permet d'avoir des informations sur le nombre de connections sur un ou des serveurs *WEB*, ainsi que sur les domaines et *IP* des internautes utilisant le site, mais encore sur les sites qui ont des liens vers notre site ou les mots clefs qui on conduit à notre site sur les moteurs de recherche comme *Google*. Pour celà, *axstats* scanne les logs d'Apache.

L'outil *awstats* a un fichier de configuration par site. Dans notre exemple, nous appelerons `default` notre site, mais on pourrait monitorer d'autres sites de manière similaire. Il faut d'abord créer le fichier `awstats.default.conf`

```
cd /etc/awstats/
cp awstats.conf awstats.default.conf
```

On édite ensuite ce fichier, qui comprend principalement les options suivantes :

- **LogFile="/var/log/apache2/access.log"** Chemon vers les logs d'Apache pour le site. Cette option doit être cohérente avec les format de logs défini dans le fichier de /etc/apache2/sites-
  qui corresponda au site.
- **LogFormat=1** : le format des logs d'Apache. La valeur 1 correspond au format combined. Cette option doit être cohérente avec les format de logs défini dans le fichier de /etc/apache2/sites-
  qui corresponda au site.
- **SiteDomain="my_domain"** : Le domaine di site à monitorer.
- **HostAliases="localhost 127.0.0.1 193.56.34.1 www.my_domain.com"** : doit contenir tous les surnoms de notre nom de domaine.
- **DNSLookup=1** : pour activer la recherche des noms de domaines des clients du site par des requêtes DNS. A éviter sur les sites très actifs. Mettre la valeur 2 pour ne pas faire de recherche DNS.
- **DirData="/chemin/vers/awstats-data"** : chemin vers le répertoire dans lequel les données générées par *axstats*, et notamment la base de données, seront stockées.
- **DirCgi="/cgi-bin"** : répertoire dans lequel nous mettrons le script *PERL* awstats.pl si l'on souhaite mettre à jour et consulter les stats à distance.

On met ensuite les scripts *Perl* awstats.pl et awstats_buildstaticpages.pl (cherchez avec find) dans le répertoire /cgi-bin (voir configuration du ScriptAlias d'Apache). Celà permet de mettre à jour les stats via le web.

```
# met à jour la base de données awstats :
nice perl /chemin/cgi-bin/awstats.pl -update
# construit les pages HTML contenant les résultats
nice perl /chemin/cgi-bin/awstats_buildstaticpages.pl \
     awstatsprog=/home/httpd/cgi-bin/awstats.pl \
     -config=default -dir=/chemin/où/mettre/les/html/
```

Pour consulter, dans votre navigateur, tapez l'URL vers le fichier *HTML* awstats.default.html dans les répertoire donné à awstats_buildstaticpages.pl, soit en local sur le serveur, soit à partir d'une station distante dûment autorisée dans la configuration d'Apache.

Pour bien faire, il faut mettre à jour la base de données d'*awstats* régulièrement car les logs d'Apache net restents pas très longtemps. On peut créer un *cronjob* avec crontab -e.

Chapitre 6

Configurer un serveur *FTP* avec `proftp`

FTP est un protocole d'échange de fichiers. Le serveur indique quels sont les répertoires partagées et gère les mots de passe.

Un client `ftp` peut se connecter à un serveur en donnant le nom de la machine serveur, en donnant un login et mot de passe.

Sous linux, le serveur le plus couramment utilisé est *wu-ftp*. Or celui-ci est très peu sécurisé et laisse passer les mots de passe en clair.

ProFTP est un serveur `ftp` demandant moins de ressources et offrant une sécurité accrue. Les mots de passe sont cryptés par défaut (si le client le supporte !), les données peuvent l'être grâce à un système de certificats. De plus, la gestion des droits d'accès et la configuration sont proches de celles d'apache. Il faut cependant être conscient qu'*FTP* est loin d'offrir les mêmes condition de sécurité qu'*SSH* ou *SFTP*.

On démare `proftpd` par la commande :

```
/etc/init.d/proftpd start
```

On relance `proftpd` après un changement de configuration par la commande :

```
/etc/init.d/proftpd restart
```

On arette `proftpd` par la commande :

```
/etc/init.d/proftpd stop
```

6.1 Fichier de configuration proftpd.conf

Nous devons à présent passer à la configuration du serveur. Ouvrez le fichier

```
/etc/proftpd/proftpd.conf
```

Notez que sa syntaxe est très proche de celle du fichier de configuration d'apache. Voici les commandes de configuration principales :

- **`ServerName`** : indique le nom du serveur qui s'affichera vers le client.
- **`User`** et **`Group`** : l'utilisateur sous l'*UID* duquel le serveur tournera (par sécurité, il vaut mieux ne pas mettre `root` mais un utilisateur avec peu de privilèges).

- **ServerType** standalone : signifie que le serveur reste en écoute du réseau. Lorsqu'il recoit une demande de connexion, il crée un processus fils et se remet en écoute. On peut aussi utiliser inet, ce qui permet de laisser l'écoute au démon inetd(TCP_WRAPPER)

- **Umask** 022 : la valeur 022 permet d'interdire la création d'un nouveau fichier par un accès en écriture ; seule la mise à jour d'un fichier existant est autorisée.

- Déclaration d'un virtual host : par exemple pour un virtual host GLOBAL :

```
<GLOBAL>
   <LIMIT...>
      # mettre ici des restrictions d'accès
   </LIMIT>
   etc...

</GLOBAL>
```

- **DisplayLogin** fichier indique le nom du fichier qui donne un message de bienvenue. dans le message %U indique le nom de l'utilisateur qui s'est connecté, %R le nom d'hôte du client, %T la date (heure du serveur)...

- **<Limit Commande > DenyAll </Limit>** Placé dans un virtual host, refuse l'utilisation de commandes par les utilisateurs se connectant au serveur ftp.
  Les commandes peuvent être READ, WRITE, LOGIN, MKD RNFR RNTO DELE RMD STOR CHMOD SITE_CHMOD SITE XCUP XRMD PWD XPWD,....
  Les commandes plus utilisées sont READ, WRITE, et LOGIN.
  Les permissions sont similaires aux drois d'accès aux répertoires d'apache : Allow All, Deny All. On peut par exemple limiter l'accès à certains utilisateurs :

```
<Limit LOGIN >
   AllowUser toto
   DenyUser badguy
</Limit>
```

  ou encore interdire l'accès en écriture à tous :

```
<Limit WRITE>
   DenyAll
</Limit>
```

- **MaxInstances** 30 : limite le nombre de processus simultanés autorisés avec les identifiants de groupe et d'utilisateur considéré. **ExtendedLog** /var/log/ftp.log : spécifie le nom de fichier log

- **AllowOverwrite** on : autorise un utilisateur d'écraser un fichier qui lui appartient.

- **UseFtpUsers** on : définit dans le fichier /etc/ftpusers les utilisateur qui n'ont pas accès au serveur ftp. Par exemple, il faut ajouter anonymous pour interdire l'accès ftp anonyme.

- **`DefaultChdir`** `/var/ftp` Indique le répertoire par défaut du serveur. Les utilisateurs se trouvent placés dans ce répertoire lors de la connexion. **`DefaultRoot`** `/var/ftp` : déclare ce répertoire comme la racine du système de fichiers.
- **`UserRatio`** `toto N...` permet la gestion des ratios. Permet de controler la quantité de fichiers et d'octets que les utilisateurs sont autorisés à transférer.
- **`SaveRatios`** `1` : sert à préciser que nous souhaitons sauvegarder les crédits de chaque utilisateur entre 2 sessions.

6.2 Exemple de fichier `proftpd.conf`

```
#
# /etc/proftpd/proftpd.conf -- This is a basic ProFTPD configuration file.
# To really apply changes reload proftpd after modifications.
#

# Includes DSO modules
Include /etc/proftpd/modules.conf

# Set off to disable IPv6 support which is annoying on IPv4 only boxes.
UseIPv6                         on

ServerName                      "Debian"
ServerType                      standalone
DeferWelcome                    off

MultilineRFC2228                on
DefaultServer                   on
ShowSymlinks                    on

TimeoutNoTransfer               600
TimeoutStalled                  600
TimeoutIdle                     1200

DisplayLogin                    welcome.msg
DisplayFirstChdir               .message
ListOptions                     "-l"

DenyFilter                      \*.*/

# Port 21 is the standard FTP port.
Port                            21

# In some cases you have to specify passive ports range to by-pass
# firewall limitations. Ephemeral ports can be used for that, but
# feel free to use a more narrow range.
```

```
# PassivePorts                  49152 65534

# To prevent DoS attacks, set the maximum number of child processes
# to 30.  If you need to allow more than 30 concurrent connections
# at once, simply increase this value.  Note that this ONLY works
# in standalone mode, in inetd mode you should use an inetd server
# that allows you to limit maximum number of processes per service
# (such as xinetd)
MaxInstances                    30

# Set the user and group that the server normally runs at.
User                            proftpd
Group                           nogroup

# Umask 022 is a good standard umask to prevent new files and dirs
# (second parm) from being group and world writable.
Umask                           022  022
# Normally, we want files to be overwriteable.
AllowOverwrite                  on

# Uncomment this if you are using NIS or LDAP to retrieve passwords:
# PersistentPasswd              off

# Be warned: use of this directive impacts CPU average load!
#
# Uncomment this if you like to see progress and transfer rate with ftpwho
# in downloads. That is not needed for uploads rates.
# UseSendFile                   off

TransferLog /var/log/proftpd/xferlog
SystemLog   /var/log/proftpd/proftpd.log

<IfModule mod_tls.c>
TLSEngine off
</IfModule>

<IfModule mod_quota.c>
QuotaEngine on
</IfModule>

<IfModule mod_ratio.c>
Ratios on
</IfModule>


# Delay engine reduces impact of the so-called Timing Attack described in
# http://security.lss.hr/index.php?page=details&ID=LSS-2004-10-02
```

```
# It is on by default.
<IfModule mod_delay.c>
DelayEngine on
</IfModule>

<IfModule mod_ctrls.c>
ControlsEngine        on
ControlsMaxClients    2
ControlsLog           /var/log/proftpd/controls.log
ControlsInterval      5
ControlsSocket        /var/run/proftpd/proftpd.sock
</IfModule>

<IfModule mod_ctrls_admin.c>
AdminControlsEngine on
</IfModule>

<Global>
  DefaultChdir /home/ftp
  DefaultRoot /home/ftp
  UseFtpUsers on
  AllowForeignAddress on

  RequireValidShell            off
  <Directory /home/ftp>
    <Limit READ>
       AllowAll
     </Limit>
     <Limit LOGIN>
        AllowAll
     </Limit>
      <Limit WRITE>
         DenyAll
      </Limit>
    </Directory>
</Global>
```

6.3 Session `ftp` côté client

Au niveau d'un client, pour se connecter sur un serveur ftp, utilisez la commande

```
$ ftp nom_du_serveur
```

Une invite demande alors le login et le mot de passe. Certaines configurations de serveurs permettent un login anonyme avec `anonymous` comme login et un mot de passe vide.

Une fois connecté, vous pouvez vous déplacer de repertoires en repertoires grace à la commande *cd*. Pour récupérer un fichier utiliser la commande `get` et pour copier un fichier vers le

serveur utiliser la commande `put`. Pour clôturer la session, utilisez la commande `quit`.

Il existe de nombreuses autres commandes. Tapez `help` pour avoir la liste des commandes et `help cmd` pour avoir la description d'une commande `cmd`.

Chapitre 7

SSL et TLS

Pour la communication sécurisée entre un serveur (web, de mail par exemple) et un client sur internet, la connection doit être cryptée. La plupart des systèmes actuels reposent sur le cryptage asymétrique, ou cryptographie à clé plublique, dont RSA est un paradime phare. La faiblesse de ces systèmes repose dans la nécessité pour le destinataire du message de communiquer la clef publique à l'expéditeur du message. Lorsque l'expéditeur du message est un client web qui doit crypter ses données pour les envoyer à serveur, comment ce client peut-il s'assurer qu'il a bien affaire au bon serveur et n'utilise pas la clef publique d'un site pirate ?

7.1 Cryptographie asymétrique et l'attaque "Man In The Middle"

Un algorithme tel que RSA peut service à chiffrer des données pour qu'elles soient très difficilement déchiffrables pour tout autre que le destinataire légitime du message. Le schéma d'envoi d'un message crypté d'Alice (A) à Bob (B) avec (ou un autre schéma de cryptage asymétrique) est représenté sur la figure 7.1.

Le schéma d'envoi d'un message signé d'Alice (A) à Bob (B) avec (ou un autre schéma de cryptage asymétrique) est représenté sur la figure 7.2.

L'attaque "Man In the Middle" consiste tirer partie de la nécessité pour le destinataire du message chiffré de communiquer sa clef publique. Le "Man In The Middle" intercepte la clef publique et y substitue sa propre clé publique. Le schéma de l'attaque "Man In the Middle" lors de l'envoi d'un message crypté d'Alice (A) à Bob (B) avec (ou un autre schéma de cryptage asymétrique) est représenté sur la figure 7.3.

7.2 Sécurisation d'un serveur web par *SSL/TLS*

Le schéma d'une connexion cryptée NON SÛRE à un site web de e-commerce est représenté sur la figure 7.4.

Le schéma d'une attaque "Man In The Middle" lors d'une connexion cryptée NON SÛRE à un site web de e-commerce est représenté sur la figure 7.5.

Pour éviter l'attaque "Man In The Middle", SSL et TLS ajoutent une surcouche au protocole dans laquelle **la clef publique du serveur est signée par un algorithme de signature à clef publique**. de sorte que le client puisse établir son authenticité. Èvidemment, le serveur

ne peut pas signer lui-même sa clef publique car il faudrait la clef publique pour vérifier la signature (c'est le fameux serpent qui se mord la queue). Pour cette raison, on fait appel à une *autorité de certification* pour signer la clef publique du serveur. Il faut que l'autorité de certification, qui a en principe pignon sur rue, mette à disposition des clients potentiels sa clé publique pour que le client puisse vérifier la signature et par là l'authenticité de la clé publique du serveur. (vous me suivez ?) Le schéma d'une connexion cryptée à un site web de e-commerce suivant le protocole SSL ou TLS est représenté sur la figure 7.6.

7.3 Exemple : la génération d'un cerificat autosigné

Faire appel à une autorité de certification qui a pignon sur rue coute de l'argent et on ne le fait en général que pour rendre un site sécurisé accessible au public. En interne à une organisation, on peut utiliser des certificats autosignés, c'est à dire être soi-même sa propre autorité de certification.

Pour utiliser en production, il faudrait ajouter de l'aléatoire aux générations de clefs dans les commandes `openssl` ci-dessous. (et prendre une bonne passphrase pour la CA)

Remerciement : Davide Delon a produit les commandes `openssl` pour cette partie.

On génère un couple de clefs publique / privée pour l'autorité de certification, protégé par mot de passe

```
$ openssl genrsa -des3 -out ca.key 2048
Generating RSA private key, 2048 bit long modulus
..........+++
.................................+++
e is 65537 (0x10001)
Enter pass phrase for ca.key:**********
Verifying - Enter pass phrase for ca.key:**********
```

On fabrique un certificat pour notre authorité de certification valable 10 ans

```
$ openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
Enter pass phrase for ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]: My Certificate Authority
Organizational Unit Name (eg, section) []:Technical department
Common Name (e.g. server FQDN or YOUR name) []:www.remysprogwebtuto.org
Email Address []:remy@example.com
```

On fabrique un certificat pour notre site web avec demande de signature

```
$ openssl req -new -nodes -keyout apache.pem -out apache.csr
Generating a 1024 bit RSA private key
.........++++++
...++++++
writing new private key to 'apache.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Billom
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Remy's tech tests
Organizational Unit Name (eg, section) []:Technical tests department
Common Name (e.g. server FQDN or YOUR name) []:www.remystuto.org
Email Address []:remy@malgouyres.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:.
An optional company name []:.
```

On signe le certificat de notre site web avec notre certificat CA

```
$ openssl x509 -req -CA ca.crt -CAkey ca.key -in apache.csr -out apache.crt
Signature ok
subject=/C=FR/L=Billom/O=Remy's tech tests/OU=Technical tests department/CN=www.remystut
Getting CA Private Key
Enter pass phrase for ca.key:**********
```

On stocke les clefs publique et privée dans le même répertoire pour apache2 Attention aux droits des fichiers contenant une clef privée !

```
$ chown root.www-data apache.crt apache.csr apache.pem ca.crt ca.key ca.srl
$ chmod 640 apache.pem apache.csr
$ chmod 600 ca.key
```

Dans la réalité, ca.key est conservée précieusement sur une autre machine sécurisée !

```
$ ls -l /etc/apache2/certs
insgesamt 24
-rw-r--r-- 1 root www-data 1123 Mai 24 08:46 apache.crt
```

```
-rw-r----- 1 root www-data  725 Mai 24 08:46 apache.csr
-rw-r----- 1 root www-data 2039 Mai 24 08:46 apache.pem
-rw-r--r-- 1 root www-data 1391 Mai 24 08:45 ca.crt
-rw------- 1 root www-data 1743 Mai 24 08:43 ca.key
-rw-r--r-- 1 root www-data    3 Mai 24 08:46 ca.srl
```

On importe enfin le certificat en tant qu'autorité de certification dans le navigateur de chaque poste client.

- Dans firefox : *Outils-¿Options-¿Avancé-¿Chiffrement-¿Afficher les certificats-¿Importer...*
- Dans Chrome : *Paramètres-¿Afficher les options avancées-¿Gérer les certificats*
- Dans Internet Explorer : *Options Internet-¿Contenu-¿Certificats-¿Importer dans les CA de confiance.*

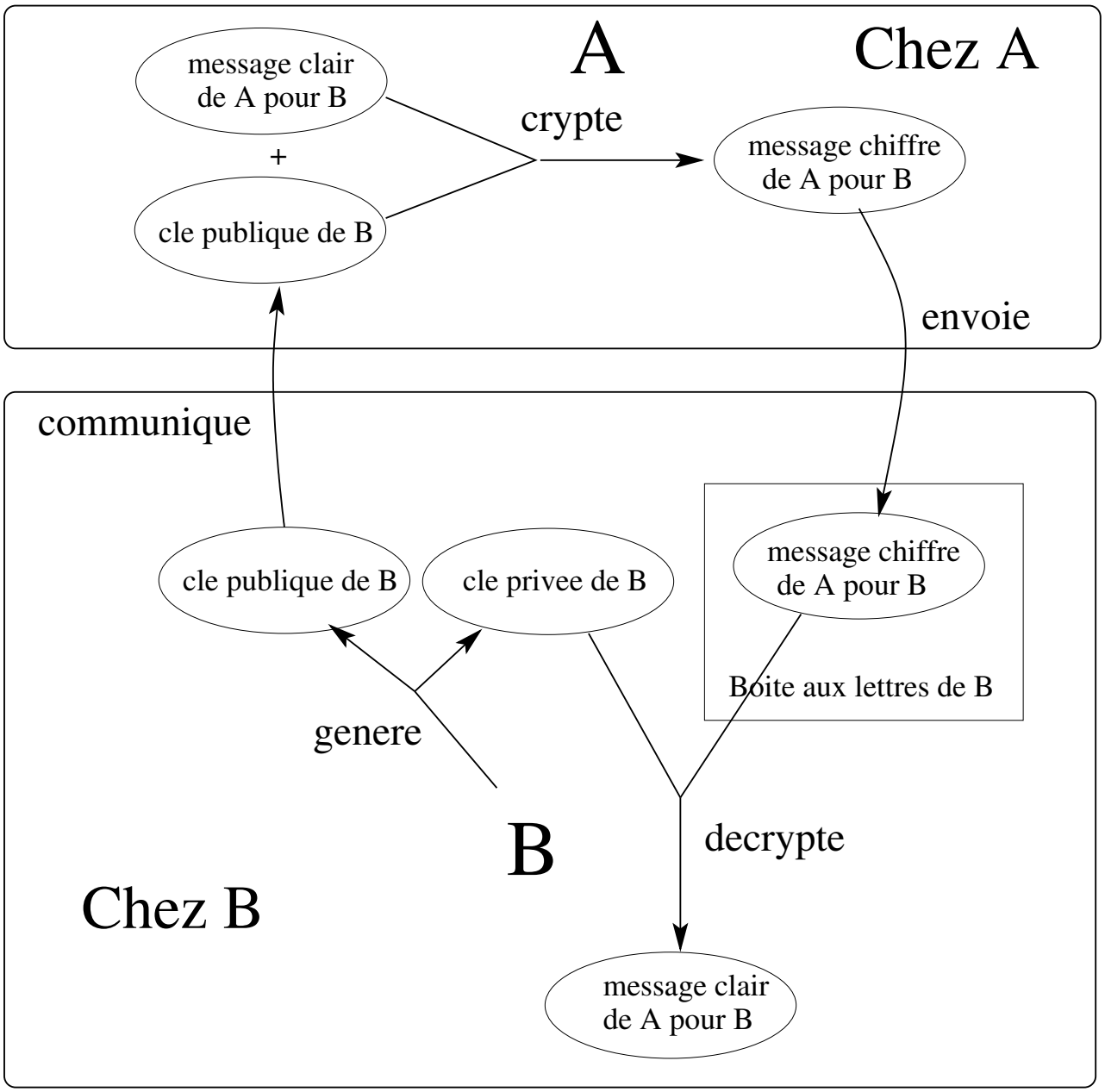


FIGURE 7.1: A envoie un message crypté à B

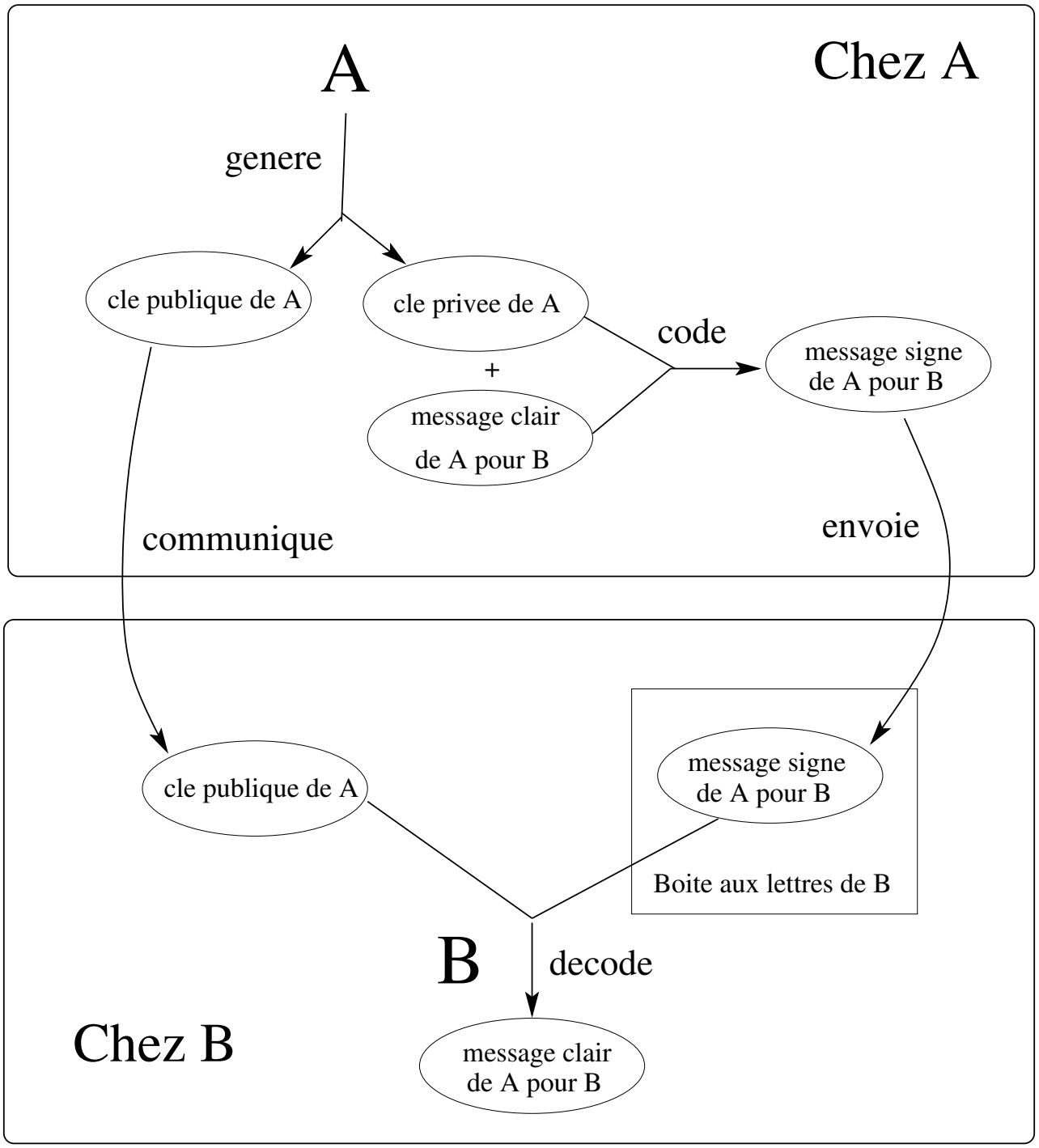


FIGURE 7.2: A envoie un message signé à B

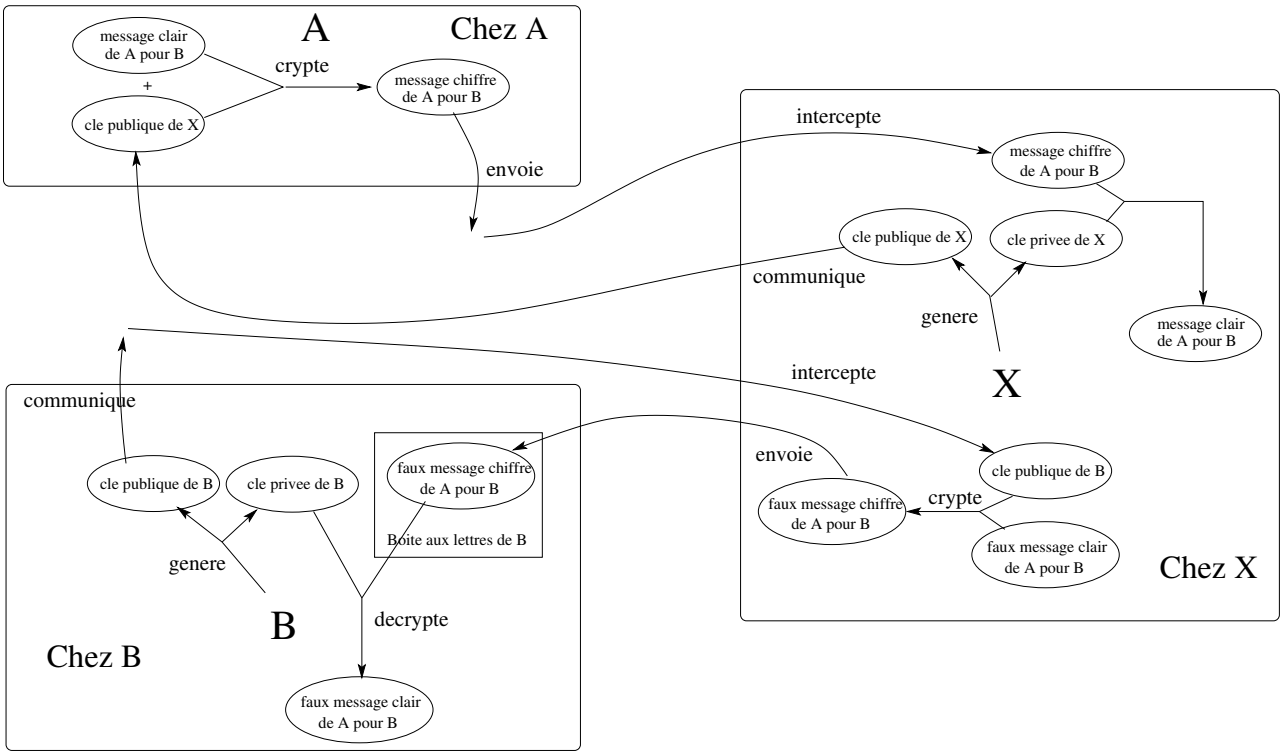


FIGURE 7.3: L'attaque dite "Man In The Middle" dans le cas d'envoi d'un message crypté.

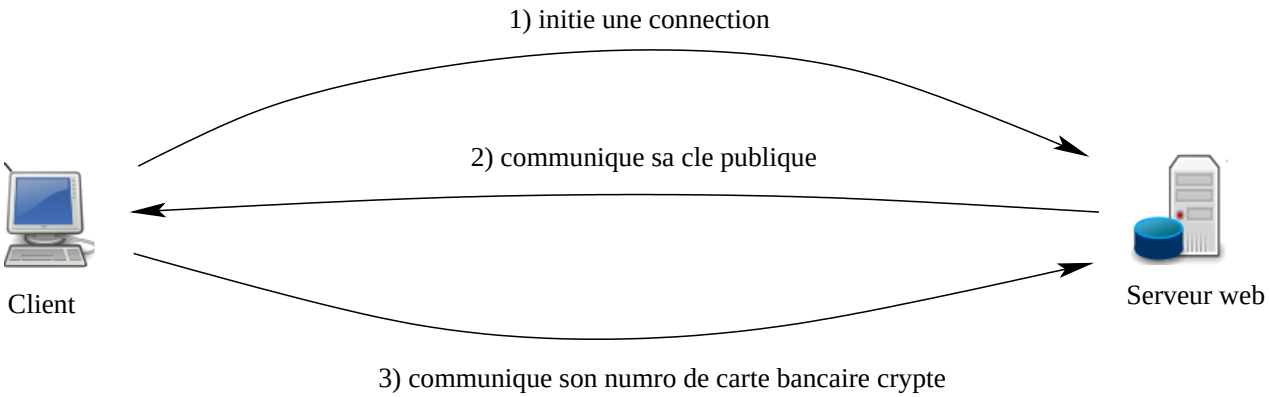


FIGURE 7.4: Idée simple et naïve d'une connexion à un site de *e*-commerce (non recommandé).

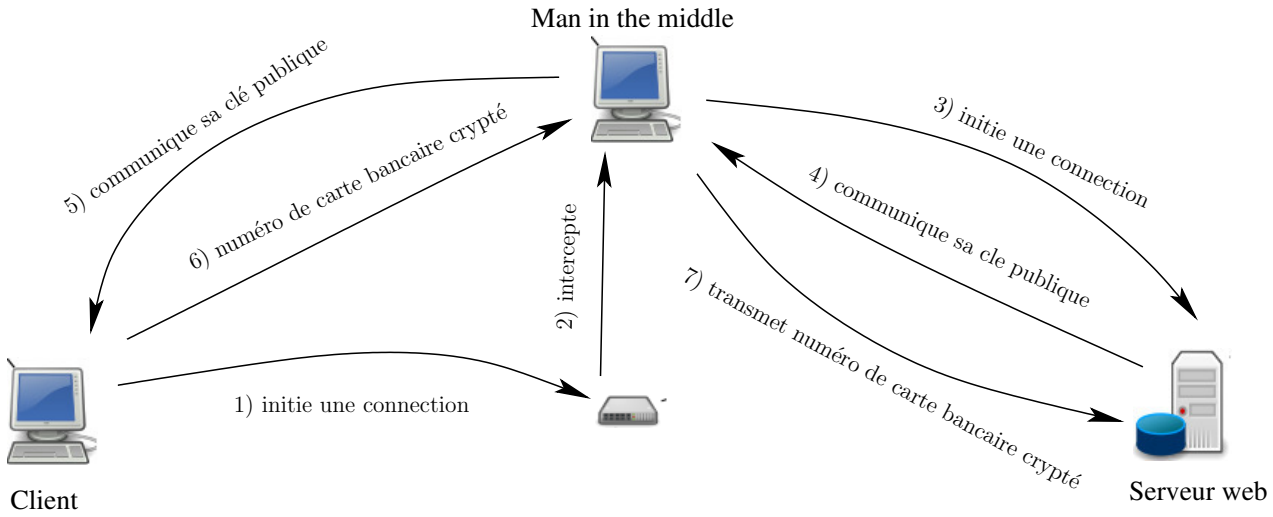


FIGURE 7.5: Application de l'attaque "Man In The Middle" à un site *e*-commerce (non recommandé).

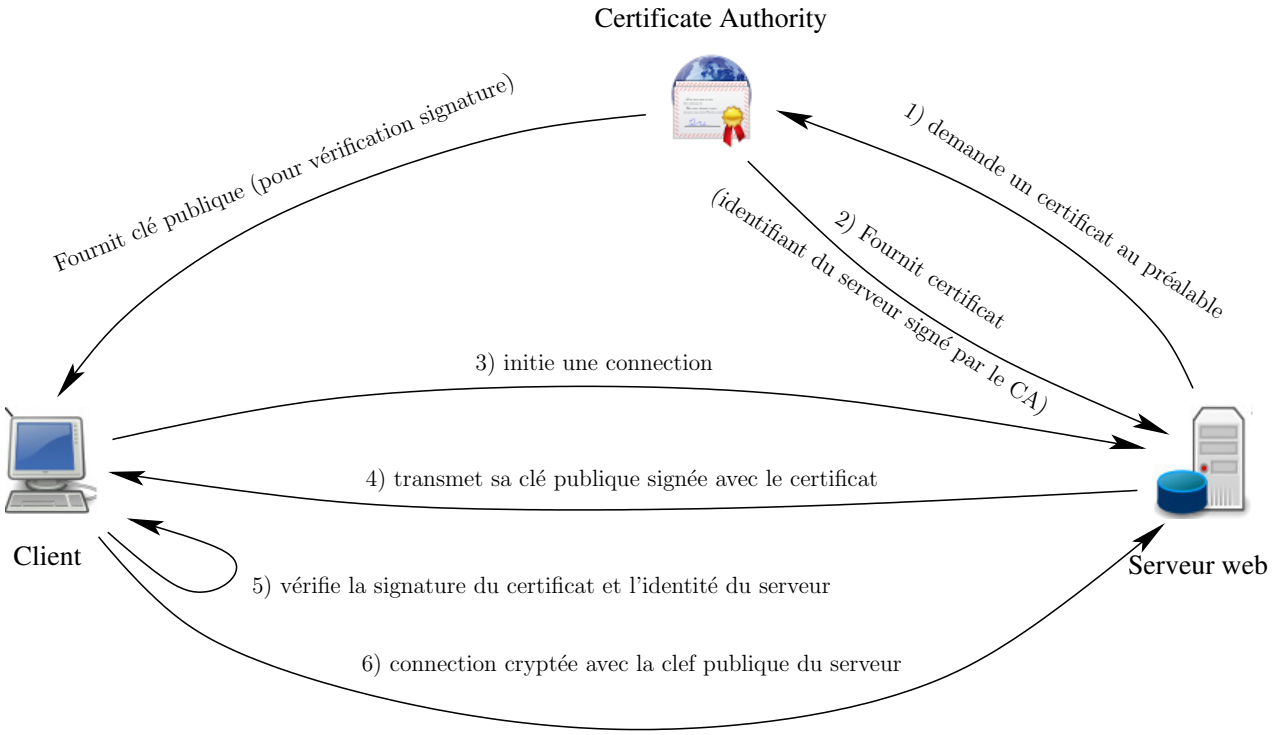


FIGURE 7.6: Principe d'une connection *SSL* ou de sa version plus récente *TLS*.

Chapitre 8

Configurer un serveur de mail avec `postfix`

8.1 Le *Mail Transfer Agent Postfix*

Un *Mail Tranfer Agent (MTA)*, ou *SMTP daemon*, permet de transmettre des messages (mail) d'un ordinateur à un autre. Pour envoyer un mail, on se connecte au serveur *SMTP* en s'autentifiant en utilisant un client de mail (outlook, thunderbird,...), qui transmet le message. Le serveur *SMTP* applique des filtres (souvent antispam) sur le mail. Le *MTA* postfix a aussi des fonctionnalités de *Mail Delivery Agent* qui lui permettent de livrer le courrier dans une mailbox (*mbox* ou *Maildir*). Les clients utilisateur utilisent ensuite un serveur *POP* pour aller chercher le mail dans la mailbox qui se trouve sur le serveur.

La configuration de base de *Postfix* se fait dans le fichier `/etc/postfix/main.cf`. On y spécifie essentiellement les filtres à appliquer au mail pour éviter que les utilisateurs ne reçoivent trop de spams, mais aussi pour éviter que le *MTA* ne soit utilisé comme relai par des spammeurs.

Voici un exemple de fichier `/etc/postfix/main.cf` :

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version

smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
delay_warning_time = 4h


myhostname = mondomaine.com

# définitions d'alias dans /etc/postfix/aliases
alias_maps = hash:/etc/postfix/aliases
alias_database = hash:/etc/postfix/aliases
myorigin = /etc/mailname
```

```
mydestination = mondomaine.com, localhost.localdomain, localhost
relayhost =

# pour une utilisation avec
# le serveur POP courier-pop
mailbox_command = /usr/bin/maildrop


mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all

# definition du reseau local privilegie
mynetworks = 127.0.0.0/8 192.168.1.0/24

smtpd_recipient_restrictions=
# règles eliminant une partie du spam
# en vérifiant la conformité des paramètres
# (expéditeur, destinataire,...) a certains protocoles
        reject_invalid_hostname,
# commenter cette ligne pour les utilisateurs d'outlook :
#       reject_non_fqdn_hostname,
        reject_non_fqdn_sender,
        reject_non_fqdn_recipient,
# vérifie que l'expediteur est bien enregistré DNS
        reject_unknown_sender_domain,
#  vérifie que le destinataire est bien enregistré DNS
        reject_unknown_recipient_domain,
# rejette certains logiciels d'envoi de mail non conformes
        reject_unauth_pipelining,
# autorise a partir du réseau local
        permit_mynetworks,
# éviter de servir de relai de spammer avec pop-before-smtp
# (nécessite l'installation de pop-before-smtp)
        check_client_access hash:/var/lib/pop-before-smtp/hosts,
# ne pas changer sans savoir pourquoi
        reject_unauth_destination,
# activer le greylisting postgrey :
# nécessite l'installation de postgrey
        check_policy_service inet:[127.0.0.1]:60000,
# politique par défaut
        permit

# blacklist antispam (voir fichier /etc/postfix/access)
smtpd_sender_restrictions =
        check_client_access hash:/etc/postfix/access
```

```
# sécurisation du smtp (éviter de servir de relai de spammer)
# test de requêtes DNS
smtpd_client_restrictions = permit_mynetworks, reject_unknown_client_hostname

# antivirus amavis
# (nécessite l'instalation d'amavis)
content_filter = smtp-amavis:[127.0.0.1]:10024

# spamass-milter configuration anti spam combine avec spamassassin
# (nécessite l'installation et configuration de
# spamassassin et spamass-milter)
milter_default_action = accept
smtpd_milters = unix:/var/run/spamass.sock


# autoriser les alias (mailng listes...)
allow_mail_to_commands = alias,forward
allow_mail_to_files = alias,forward
```

Dans le fichier `/etc/postfix/aliases`, chaque ligne définit un alias. Par exemple

```
# cat /etc/postfix/aliases

#secretariat
secretariat:emilie

#pour le webmaster
webmaster:tom,martin

#administrateur (!= root)
administrateur:david,christian
```

Dans le fichier de blacklist `/etc/postfix/access`, on trouve des expéditeurs/domaines et de directives `OK` ou `REJECT`. Après `REJECT` de façon optionnelle un code d'erreur et un texte.

```
# cat /etc/postfix/access
harrass.com  REJECT
badguy@ugly.com  REJECT
```

8.2 Antispam par `Greylisting`

Le *greylisting* consiste en ce que le serveur, à la réception d'un mail, rejette temporairement le mail pour que le serveur *SMTP* qui e expédié le mail le représente plus tard. La plupart des spammeurs n'attendent pas, d'abord pour une raison de coût, et ensuite parce que ça augmenterait le risque d'être pris en flagrant délit.

Un exemple de logiciel de greylisting utilisable avec *postfix* est *postgrey*. On ajoute la ligne suivante à la section `smtpd_recipient_restrictions` de `/etc/postfix/main.cf` :

```
        check_policy_service inet:[127.0.0.1]:60000,
```

De plus, il faut configurer *postgrey* (notamment le temps que le SMTP expéditeur doit attendre avant de représenter le mail) dans le fichier /etc/default/postgrey :

```
# cat /etc/default/postgrey
# postgrey startup options, created for Debian
# (c)2004 Adrian von Bidder <avbidder@fortytwo.ch>
# Distribute and/or modify at will.

# you may want to set
#   --delay=N   how long to greylist, seconds (default: 300)
#   --max-age=N delete old entries after N days (default: 30)
# see also the postgrey(8) manpage

POSTGREY_OPTS="--inet=127.0.0.1:60000 --delay=500 --auto-whitelist-clients=5"

# the --greylist-text commandline argument can not be easily passed through
# POSTGREY_OPTS when it contains spaces.  So, insert your text here:
#POSTGREY_TEXT="Your customized rejection message here"
```

Postgrey est pourvu d'un système d'auto-whitelist : un expéditeur qui a réussi à faire passer 5 (par défaut) mails est mis dans une whitelist et ses mails sont par la suite transmis sans délai ; celà minimise l'inconvénient du délai pour les utilisteurs. Il y a aussi possibilité de whitelister des domaines dans /etc/postgrey/whitelist_clients.local.

Les logiciels *postfix* et *postgrey* communiquent à travers une socket *TCP*, par défaut sur le port 60000.

8.3 spamassassin

Le logiciel spamassassin permet de détecter les spams, et éventuellement de les rediriger. Pour celà, spamassassin attribue un score à chaque mail. Plus le score est élevé, plus le mail a de chances d'être un spam.

Pour utiliser spamassassin, il faut l'activer dans postfix en modifiant le fichier /etc/postfix/master.c

1) Rajouter l'option -o content_filter=spamassassin au service smtp :

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args #
# ==========================================================================
smtp      inet  n       -       -       -       -       smtpd
 -o content_filter=spamassassin
```

2) Rajouter une interface à postfix.

```
# Interfaces to non-Postfix software. Be sure to examine the manual
# pages of the non-Postfix software to find out what options it wants.
#
# maildrop. See the Postfix MAILDROP_README file for details.
```

```
#

spamassassin    unix    -       n       n       -       -       pipe
        user=nobody     argv=/usr/bin/spamc -f -e
        /usr/sbin/sendmail -oi -f ${sender} ${recipient}
```

Il faut aussi configurer *spamassassin* dans le fichier /etc/spamassassin/local/cf

```
# cat /etc/spamassassin/local.cf
# This is the right place to customize your installation of SpamAssassin.
#
# See 'perldoc Mail::SpamAssassin::Conf' for details of what can be
# tweaked.
#
# Only a small subset of options are listed below
#
###########################################################################

#   Add *****SPAM***** to the Subject header of spam e-mails
#
rewrite_header Subject *****SPAM*****


#   Save spam messages as a message/rfc822 MIME attachment instead of
#   modifying the original message (0: off, 2: use text/plain instead)
#
#report_safe 1


#   Set which networks or hosts are considered 'trusted' by your mail
#   server (i.e. not spammers)
#
trusted_networks 193.49.118.0/24.


#   Set file-locking method (flock is not safe over NFS, but is faster)
#
# lock_method flock


#   Set the threshold at which a message is considered spam (default: 5.0)
#
required_score 5.0


#   Use Bayesian classifier (default: 1)
#
use_bayes 1
```

```
#   Bayesian classifier auto-learning (default: 1)
#
bayes_auto_learn 1


# Enable or disable network checks
skip_rbl_checks         0
use_razor2              1
use_dcc                 1
use_pyzor               1
#
# # Mail using languages used in these country codes will not be marked
# # as being possibly spam in a foreign language.
ok_languages            all

# Mail using locales used in these country codes will not be marked
# as being possibly spam in a foreign language.
ok_locales              all

#   Set headers which may provide inappropriate cues to the Bayesian
#   classifier
#
# bayes_ignore_header X-Bogosity
# bayes_ignore_header X-Spam-Flag
# bayes_ignore_header X-Spam-Status
```

Enfin, pour plus d'efficacité, il faut charger des modules dans `/etc/spamassassin/v310.pre`. Pour activer un module, décommentez la ligne correspondante. Chaque module correspond à une technique de détection de spam qui fait augmenter le score des mails.

8.4 Milter antivirus *amavis*

Le logiciel *amavis* est un antivirus qui va scanner les pièces jointes aux mails et éliminer certains contenus.

Pour l'utiliser, il faut rajouter la ligne suivante au fichier `/etc/postfix/main.cf`

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

Il faut aussi rajouter une interface à postfix dans le fichier `/etc/postfix/master.cf` (en n'oubliant pas de personaliser le réseau local) :

```
# Interfaces to non-Postfix software. Be sure to examine the manual
# pages of the non-Postfix software to find out what options it wants.
#
# maildrop. See the Postfix MAILDROP_README file for details.
#
```

```
smtp-amavis     unix    -       -       n       -       2       smtp
        -o smtp_data_done_timeout=1200
        -o disable_dns_lookup=yes

127.0.0.1:10025 inet    n       -       n       -       -       smtpd
        -o content_filter=
        -o local_recipient_maps=
        -o relay_recipient_maps=
        -o smtpd_restriction_classes=
        -o smtpd_client_restrictions=
        -o smtpd_helo_restrictions=
        -o smtpd_sender_restrictions=
        -o smtpd_recipient_restrictions=permit_mynetworks,reject
# ne pas oublier de personaliser le réseau local
        -o mynetworks=127.0.0.0/8 192.168.1.0/24
        -o strict_rfc821_envelopes=yes
```

8.5 *spamass-milter*

Le logiciel *spamass-milter* est un milter qui permet d'éliminer certains mails qui sont des spams. spamassassin est un antispam qui attribue un score aux mails suivant leurs chances d'être des spams. Plus le score est élevé, plus le mail a de chances d'être un spam. On peut mettre spamass-milter en sortie de spamassassin pour rejeter les mails qui ont un score élevé dans spamassassin. Il vaut mieux mettre un seuil élevé (¿15 par exemple) pour éviter tout faux positifs (il n'est pas trop grave d'accepter un faux négatif : l'utilisateur reçoit un spam, mais c'est très génant de rejeter un faux positifs : un vrai mail ne parvient pas au client)

8.5.1 Configuration

a) main.cf Dans le fichier de configuration de postfix /etc/postfix/main.cf , ajoutez les lignes :

```
# spamass-milter configuration
# l'option accept est plus sûre pour ne pas perdre de mails
milter_default_action = accept
smtpd_milters = unix:/var/run/spamass.sock

# change to this for use as a non_smtpd_milter :
#non_smtpd_milters = unix:/var/run/spamass.sock
```

Les données sont transmises du démon de spamassassin (appelé spamd) à spamass-milter via une socket. Cette socket est transmise suivant le protocole unix (sur la machine locale) via un fichier physique de type socket ici nommé spamass.sock.

8.5.2 /etc/default/spamass-milter

Voici le fichier /etc/default/spamass-milter

```
# spamass-milt startup defaults

# OPTIONS are passed directly to spamass-milter.
# man spamass-milter for details

# Default, use the nobody user as the default user, ignore messages
# from localhost

# this one is the right one for use a smtpd_milters
# Reject emails with spamassassin scores > 15 : option -r 15
# Do not modify Subject:, Content-Type: or body option -m
OPTIONS="-u nobody -m -r 15 -i 127.0.0.1 -f -p /var/spool/postfix/var/run/spamass.sock"

SOCKET="/var/spool/postfix/var/run/spamass.sock"
SOCKETOWNER="postfix:postfix"
SOCKETMODE="0660"
```

8.5.3 Création de la socket

Il faut créer une première fois les sockets dans le système de fichiers (car la socket n'est pas créée lors de l'appel automatique de spamas-milter par postfix).

```
mkdir /var/spool/postfix/var
mkdir /var/spool/postfix/var/run
```

puis créer les sockets :

```
# spamass-milter -m -u nobody -f -p /var/run/spamass.sock
# chown postfix.users /var/run/spamass.sock
# spamass-milter -m -u nobody -f -p /var/spool/postfix/var/run/spamass.sock
# chown postfix.users /var/spool/postfix/var/run/spamass.sock
```

(si j'ai bien compris, postfix tourne en chrooté et pas spamass-milter et il faut créer les deux sockets à deux endroits différents pour qu'il y ait communication entre les deux sockets. feedback wellcome)

Comme il faut faire ces quatres dernières commandes à chaque reboot, on peut les mettre dans un script (par exemple) appelé /etc/init.d/initspamass et faire lancer ce script au démarage

```
# chown +x /etc/init.d/initspamass
# update-rc.d initspamass defaults 99
```

Il faut mettre 99 pour être sûr que ce script est exécuté après le démon /etc/init.d/spamass-milter au démarage.

8.5.4 Vérification dans les logs

```
# grep Milter /var/log/mail.log
# grep milter /var/log/mail.log
May 27 08:36:46 laic spamass-milter[29641]: spamass-milter 0.3.1 starting
#
```

Troisième partie

Routage, firewall

Chapitre 9

Routage

9.1 Adresses *IP* et *MAC*

Chaque interface de chaque ordinateur sera identifié par

- Son adresse *IP* : une adresse *IP* (version 4, protocole *IPV4*) permet d'identifier un hôte et un sous-réseau. L'adresse *IP* est codée sur 4 octets. (les adresses *IPV6*, ou *IP* next generation seront codées sur 6 octets).

- L'adresse mac de sa carte réseau (carte ethernet ou carte wifi) ;

Une adresse *IP* permet d'identifier un hôte. Une passerelle est un ordinateur qui possède plusieurs interfaces et qui transmet les paquets d'une interface à l'autre. La passerelle peut ainsi faire communiquer différents réseaux. Chaque carte réseau possède une adresse *MAC* unique garantie par le constructeur. Lorsqu'un ordinateur a plusieurs plusieurs interfaces, chacune possède sa propre adresse *MAC* et son adresse *IP*. On peut voir sa configuration réseau par `ifconfig`.

```
$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:B2:3A:24:F3:C4
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2c0:9fff:fef9:95b0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:5
          collisions:0 txqueuelen:1000
          RX bytes:1520 (1.4 KiB)  TX bytes:2024 (1.9 KiB)
          Interrupt:10
```

On voit l'adresse *MAC* `00:B2:3A:24:F3:C4` et l'adresse *IP* `192.168.0.2`. Cela signifie que le premier octet de l'adresse *IP* est égal à 192, le deuxième 168, le troisème octet est nul, et le quatrième vaut 2.

9.2 Sous-réseaux

9.2.1 Classes de sous-réseaux

Les addresses *IPV4* sont organisées en sous-réseaux. Chaque sous-réseau possède une adresse, qui est une partie de l'adresse *IP* des machines de ce sous-réseau.

Par exemple, l'adresse *IP* `192.168.4.35` appartient au sous-réseau `192.168.4`, parfois aussi noté `192.168.4.0`.

Les sous-réseaux sont organisés en *classes*. Chaque classe de sous-réseaux correspond à des réseaux pouvant contenir un certain nombre de machines.

- **Classe *A* :** les adresses de de `1.0.0.0` à `127.0.0.0`.

  L'identifiants du réseau est alors sur 8 bits et les identifiants de machine sont sur 24 bits (plusieurs millions de machines par sous-réseau ;

- **Classe *B* :** les adresses de de `128.0.0.0` à `191.255.0.0`. L'identifiants du réseau est alors sur 16 bits et les identifiants de machine sont sur 16 bits (plus de 65000 machines par sous-réseau) ;

- **Classe *C* :** les adresses de de `192.0.0.0` à `223.255.255.0`. L'identifiants du réseau est alors sur 24 bits et les identifiants de machine sont sur 8 bits (au plus 254 machines par sous-réseau, numérotées de 1 à 254) ;

9.2.2 Masque de sous-réseau

Un masque de sous-réseau est une donnée sur 4 octets qui, avec l'adresse du sous-réseau, caractérise les *IP* du sous-réseau. Un bit du masque de sous-réseau est à 1 si pour toutes les adresses *IP* du sous-réseau, ce même bit est le même pour l'adresse *IP* et le sous-réseau.

Par exemple, pour le réseau de classe *A* `37.0.0.0` avec le masque de sous-réseau `255.0.0.0`, les 8 premiers bits de toutes les adresses *IP* du sous-réseau valent 37.

Autre exemple : pour le sous-réseau de classe *C* `193.56.49.0` et le masque de sous-réseau `255.255.255.0`, les 24 premiers bits de toutes les *IP* du sous-réseau sont `193.56.49`.

On peut désigner un sous-réseau par son adresse et son masque, mais on peut aussi désigner le sous-réseau en donnant seulement le nombre de bits du masque. On parle alors, pour reprendre les deux exemples précédents, du sous-réseau `37.0.0.0/8` ou du sous-réseau `193.56.49.0/24`.

9.3 Routage

Le routage permet de faire communiquer plusieurs sous-réseaux. Une *passerelle* (en anglais *gateway*) est en communication avec différents sous-réseaux sur différentes interfaces, et assure la communication entre les différents sous-réseaux (voir figure 9.1).

9.3.1 Routes

Une *route* définie sur une station est un chemin que doivent prendre les paquets à destination d'un certain sous-réseau. Prenons l'exemple (voir figure 9.1) d'une station, appelée *station* 1,

d'adresse *IP* `121.65.77.8` sur un réseau `112.0.0.0/8`. Elle est connectée à une passerelle qui a pour *IP* dans ce réseau `112.65.123.3` sur son interface `eth0`. La passerelle est aussi connectée au réseau `192.168.0.0/24` via son interface `eth1` qui a pour *IP* `192.168.0.1`. Si la station 1

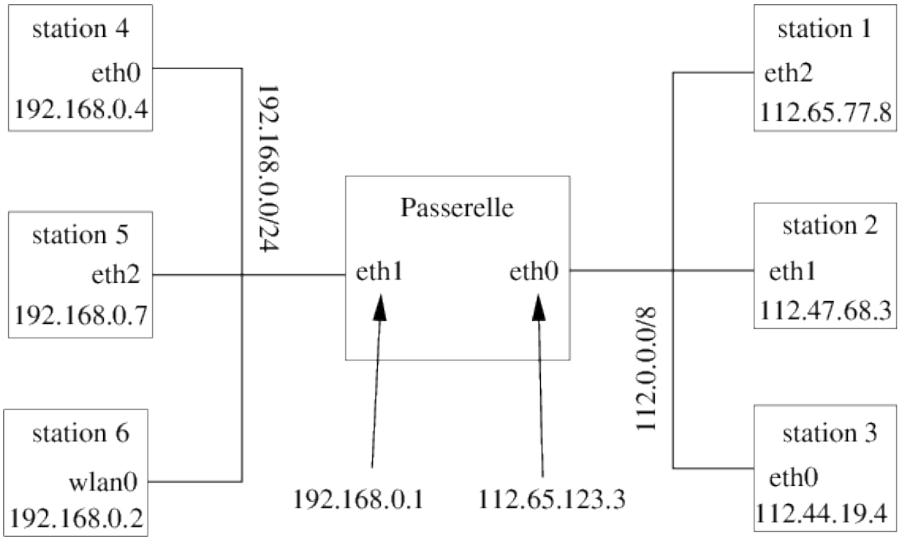


FIGURE 9.1: Exemple de passerelle faisant communiquer deux réseaux

veut communiquer directement avec la station, appelée *station* 6, d'adresse *IP* `192.168.0.2` sur le réseau `192.168.0.0/24`, trois condition doivent être réunies ;

- Une route doit être définie sur la station 1 indiquant que les paquets à destination du réseau `192.168.0.0/24` doivent passer par la passerelle `112.65.123.3`. Pour celà, on peut utiliser la commande `route` :

  ```
  # route add -net 192.168.0.0/24 gw 112.65.123.3
  ```

- Une route doit être définie sur la station 6 indiquant que les paquets à destination du réseau `112.0.0.0/8` doivent passer par la passerelle `192.168.0.1` ; Pour celà, on peut utiliser la commande `route` :

  ```
  # route add -net 112.0.0.0/8 gw 192.168.0.1
  ```

- La passerelle doit être configurée pour transmettre (ou *forwarder*) les paquets *IP* d'un réseau à l'autre, ce qui se fait par la commande :

  ```
  # echo 1 >/proc/sys/net/ipv4/ip_forward
  ```

Remarque 9.3.1 Il faut refaire ces commandes après un reboot. Pour éviter de relancer ces commandes manuellement, on peut les mettre dans des scripts d'initialisation au démarage avec la commande `update-rc.d` (sous *debian*)). Pour ajouter une script `my_ script` à l'initialisation :

```
# mv my_script /etc/init.d
# update-rc.d my_script defaults
```

On peut voir l'état des routes avec la commande `route -n`. Par exemple, sur la station 1 :

```
# route -n
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     112.65.123.3    255.255.255.0   U     0      0        0 eth2
etc...
```

Sur la station 6

```
# route -n
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
112.0.0.0       192.168.0.1     255.0.0.0       U     0      0        0 wlan0
etc...
```

Pour supprimer une route, par exemple vers le réseau `193.86.46.0/24` via une passerelle `196.24.52.1`, on fait :

```
# route del -net 193.86.46.0/24 gw 196.24.52.1
```

9.3.2 Route par défaut (*gateway*)

Quand on a défini un certain nombre de routes sur une station, on peut définir une route spéciale pour les paquets *IP* à destination des réseaux non prévus dans les autres routes. On appelle une telle route une *route par défaut*. En général, c'est la route qu'il faut employer pour aller sur internet. On emploie le réseau `0.0.0.0` (masque `255.255.255.255`). Pour définir une route par défaut on peu employer `route`. Par exemple, pour définir la route par défaut via la passerelle `194.56.87.1` :

```
route add default gw 194.56.87.1
```

Pour supprimer cette même route :

```
route del default gw 194.56.87.1
```

9.3.3 *NAT* et *masquerading*

Lorqu'un hôte ayant une adresse *IP* sur un réseau local cherche à se connecter sur un réseau plus vaste, par exemple sur internet, via une passerelle, cet hôte a besoin d'une adresse *IP* sur le réseau vaste. Pour celà, soit on demande à ce que les adesses du réseau local soient routées sur le réseau global, mais il faut alors demander à réserver un plage d'adresses sur le réseau global, soit l'administrateur de la passerelle a la possibilité de prêter l'*IP* de la passerelle aux machines du réseau local. Pour cela, on utilise *iptables* avec *NAT*. Par exemple, si la passerelle se connecte à internet via son interface `eth0`, il suffit d'exécuter la commande suivante sur la passerelle :

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Toute machine du réseau local (par exemple `192.168.0.0/24`) qui se connecte à internet via cette passerelle aura alors l'adresse *IP* de la passerelle sur internet. On peut aussi donner aux machines du réseau local une autre adresse *IP* que l'on spécifie avec `--to` :

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 193.56.17.9
```

Nous aurons un apperçu plus complet des possibilités d'*iptables* au chapitre 11.

Chapitre 10

Protocoles, services, ports

10.1 La listes des protocles connus du systèmes

Un protocole (*IP*, *TCP*, *UDP*,...) est un mode de communication réseau, c'est à dire une manière d'établir le contact entre machine et de transférer les données. Sous linux, la liste des protocoles reconnus par le système se trouve dans le fichier `/etc/protocols`.

```
$ cat /etc/protocols
# Internet (IP) protocols
ip      0       IP              # internet protocol, pseudo protocol number
#hopopt 0       HOPOPT          # IPv6 Hop-by-Hop Option [RFC1883]
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ‘‘IP’’)
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
igp     9       IGP             # any private interior gateway (Cisco)
pup     12      PUP             # PARC universal packet protocol
udp     17      UDP             # user datagram protocol
hmp     20      HMP             # host monitoring protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
rdp     27      RDP             # "reliable datagram" protocol
etc...          etc...
```

A chaque protocole est associé un numéro d'identification standard. Le protocle *IP* est rarement utilisé directement dans une application et on utilise le plus couramment les protocoles *TCP* et *UDP*.

10.1.1 Le protocole *TCP*

Le protocole *TCP* sert à établir une communication fiable entre deux hôtes. Pour cela, il assure les fonctionnalités suivantes :

- Connexion. L'émetteur et le récepteur se mettent d'accord pour établir un connexion. La connexion reste ouverte jusqu'à ce qu'on la referme.
- Fiabilité. Suite au transfert de données, des tests sont faits pour vérifier qu'il n'y a pas eu d'erreur dans la transmission. De plus, les données arrivent dans l'ordre où elles ont été émises.
- Possiblilité de communiquer sous forme de flot de données, comme dans un tube (par exemple avec les fonctions `read` et `write`).

10.2 Services et ports

Il peut y avoir de nombreuses applications réseau qui tournent sur la même machine. Les numéros de port permettent de préciser avec quel programme nous souhaitons dialoguer par le réseau. Chaque application qui souhaite utiliser les services de la couche *IP* se voit attribuer un uméro de port. Un numéro de port est un entier sur *16* bits (deux octets). Dans un programme *C*, on peut stocker un numéro de port dans un `unsigned short`. Il y a un certain nombre de ports qui sont réservés à des services standards. Pour connaître le numéro de port correspondant à un service tel que `ssh`, on peut regarder dans le fichier `/etc/services`.

```
# Network services, Internet style
tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
```

```
tacacs          49/tcp                          # Login Host Protocol (TACACS)
tacacs          49/udp
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
etc...
```

L'administrateur du système peut définir un nouveau service en l'ajoutant dans `/etc/services` et en précisant le numéro de port. Les numéros de port inférieurs à 1024 sont réservés aux serveurs et démons lancés par root (éventuellement au démarage de l'ordinateur), tels que le serveur d'impression `/usr/sbin/cupsd` sur le port ou le serveur `ssh /usr/sbin/sshd` sur le port 22.

Chapitre 11

Firewall configurés avec *iptables*

Le pare-feu (en anglais *firewall*) *netfilter*, configuré avec la commande *iptables*, permet de filtrer les paquets réseau entrants, sortants, et transmis, sur une machine. On peut filtrer par interface, par port, par adresse de source ou de destination des paquets. La configuration d'*iptables* permet aussi de partager une adresse *IP* (par exemple sur internet) entre plusieurs machines d'un réseau local.

11.1 Principe d'*iptables*

11.1.1 Les politiques

Lorsqu'un paquet *IP* rentre ou sort de votre machine, on peut adopter trois politiques différents : `ACCEPT`, `REJECT` ou `DROP`. Avec la politique `ACCEPT`, le paquet est simplement transmis normalement. Avec la politique `REJECT` le paquet n'est pas transmis et la machine source est prévenue. Avec la politique `DROP`, je paquet n'est pas transmis et la machine source n'est pas prévenue.

11.1.2 Les directions de paquets

Il y a principalement trois directions de paquets (ces directions de paquets sont appelés chaînes) qui circulent sur la machine :

1. `INPUT` : paquets entrants à destination de la machine et venant d'une autre machine ;
2. `OUTPUT` : paquets sortants venant de la machine et à destination d'un autre machine ;
3. `FORWARD` paquets venant d'un autre machine et à destination d'une troisième machine lors de l'utilisation de la machine comme passerelle pour le routage.

11.1.3 Les rêgles

La configuration *iptables* consiste en un ensemble de règles. Une règle est une commande *iptables* en *bash* comprenant :

- `-A chaine` ou `-I chaine` : La chaîne `INPUT`, `OUTPUT` ou `FORWARD` ;
- `-i interface` et/ou `-o interface` Les interfaces d'entrée et de sortie (optionnel) ;

- `-p protocole` Le protocole (si besoin) (voir `/etc/protocols`) ;
- `--sport` ou `--dport` ; Les ports de source ou de destination (si besoin) ;
- `-s adresse` ou `-d adresse` : L'adresse (ou le sous-réseau) de provenance ou de destination (optionnel). On pet aussi utiliser la négation (avec `!`) pour exclure une adresse au lieu d'inclure une adresse ;
- La politique `ACCEPT`, `REJECT` ou `DROP`, spécifiée avec l'option `-j`.

L'ordre des règles est important. Si l'on met, par exemple, un politique de rejet systématique d'un certain type de paquets, on peut ensuite mettre une règles qui accepte ce type de paquet à partir d'une certaine machine ou d'un certain sous réseau. C'est la différence entre `-A` et `-I`. Le `-A` rajoute la règle à la fin d'une chaîne et le `-I` au début.

Pour vider toutes les règles :

```
iptables -F chaîne
```

Pour mettre une politique par défaut (qui s'applique à tous les paquets sauf règle contraire) :

```
iptables -P chaîne politique
```

En général, on met toutes les règles dans un script qui va créer toutes les tables. On peut aussi lancer ce script automatiquement au démarage avec `update-rc.d`.

11.2 Exemple script de configuration

```
# On vide toutes les règles
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD

# Politique par défaut (version très sécurisée)
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Autoriser le trafic sur l'interface loopback
iptables -I INPUT -i lo -j ACCEPT
iptables -I OUTPUT -o lo -j ACCEPT

# protocole ICMP (dont ping) dans tous les sens
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A FORWARD -p icmp -j ACCEPT

# ssh vers l'extérieur (client ssh) via l'interface eth0
iptables -I INPUT -p tcp -i eth0 --sport ssh -j ACCEPT
```

```
iptables -I OUTPUT -p tcp -o eth0 --dport ssh -j ACCEPT

# ssh vers la machine (serveur ssh) via l'interface eth0
iptables -I INPUT -p tcp -i eth0 --dport ssh -j ACCEPT
iptables -I OUTPUT -p tcp -o eth0 --sport ssh -j ACCEPT
```

- Pour l'utilisation d'un client (ssh, telnet,...), le `OUTPUT` a pour destination le port correspondant au service et le `INPUT` a pour source le port correspondant au service.
- Pour l'utilisation en tant que serveur (ssh, telnet,...), le `OUTPUT` a pour source le port correspondant au service et le `INPUT` a pour destination le port correspondant au service.
- D'une manière générale, les ports réservés (22 pour `ssh`, 80 pour `www`, etc...) sont utilisés coté serveur. Les ports utilisés coté client sont des ports non privilégiés numérotés de 1024 à 65535.

11.3 Le cas de `FTP`

Le service `FTP` mérite une attention particulière car après la connexion via le port 21 du serveur, le serveur demande une identification sur le port 113 du client. Après l'établissement de la connection, les données sont tansmises sur encore un autre port, ce qui permet de garder le port 21 libre pour les commandes (`put`, `get`,...) : on parle de connexion `RELATED`. (réviser son cours sur les *sockets* : lors de la connection, une nouvelle *socket* est créer qui va traîter le client dans un processus fils ou un thread). Il faut alors autoriser toutes ces connexions, y compris les connexions `RELATED` et les connexions déjà établies `ESTABLISHED`.

Voici les autorisations minimales sur un serveur *FTP* avec politique `DROP` par défaut :

```
# chargement du module du noyau spécialisé :
modprobe ip_conntrack_ftp

# autorise les connexions FTP de commande vers l'intérieur :
iptables -A INPUT -m state --state NEW -p tcp --dport 21 -j ACCEPT

# autorise les demande d'identification vers l'extérieur
iptables -A OUTPUT -m state --state NEW -p tcp --dport 113 -j ACCEPT

#  # # autorise les connexions FTP de données (ou autre connexion RELATED)
iptables -A INPUT -m state --state RELATED -j ACCEPT
iptables -A OUTPUT -m state --state RELATED -j ACCEPT

# autorise les connexions déjà établies
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

11.4 *SNAT*, *DNAT* et *masquerading*

Dans cette section, on suppose que les routes ont été établies correctement et que la passerelle a été configurée pour forwarder les paquet *IP* (voir chapitre 9).

Lorqu'un hôte ayant une adresse *IP* sur un réseau local cherche à se connecter sur un réseau plus vaste, par exemple sur internet, via une passerelle, cet hôte a besoin d'une adresse *IP* sur le réseau vaste. Pour celà, soit on demande à ce que les adesses du réseau local soient routées sur le réseau global, mais il faut alors demander à réserver un plage d'adresses sur le réseau global, soit l'administrateur de la passerelle a la possibilité de prêter l'*IP* de la passerelle aux machines du réseau local. Pour cela, on utilise *iptables* avec *NAT*. Par exemple, si la passerelle se connecte à internet via son interface `eth0`, il suffit d'exécuter la commande suivante sur la passerelle :

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Toute machine du réseau local (par exemple `192.168.0.0/24`) qui se connecte à internet via cette passerelle aura alors l'adresse *IP* de la passerelle sur internet. On peut aussi donner aux machines du réseau local une autre adresse *IP* que l'on spécifie avec `--to` :

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 193.56.17.9
```

On peut aussi changer l'adresse de destination avec *DNAT*. Celà permet de rediriger, par exemple, un accès sur un certain port vers une autre machine. Par exemple, la passerelle peut rediriger les accès *WEB* (port 80) via son interface `eth1` sur un serveur web situé sur une autre machine d'*IP* `192.168.0.5` sur le réseau local :

```
iptables -A FORWARD -p tcp --sport www -j ACCEPT
iptables -A FORWARD -p tcp --dport www -j ACCEPT
iptables -t nat -A PREROUTING -p tcp -i eth1 --dport www -j DNAT --to 192.168.0.5
```

Quatrième partie

LDAP

Chapitre 12

Protocole d'annuaire *LDAP*

LDAP (Lightweight Directory Access Protocol) est le protocole d'annuaire sur TCP/IP. Un annuaire est une base de données pouvant représenter des personnes ou des éléments d'infrastructure réseau ou autre. Le but est l'accès rapide à l'information des attributs des objets en lecture via des requêtes. Les accès en écriture sont moins optimisés. C'est ce qui différencie les services d'annuaire des système de gestion de bases de données (SGBD) traditionnels. Pour celà, les données de l'annuaire sont organisées de manière arborescente.

12.1 Le Directory Information Tree (DIT) et les entrées

Les données de l'annuaire sont stockées dans un arbre appelé *Directory Information Tree (DIT)*.

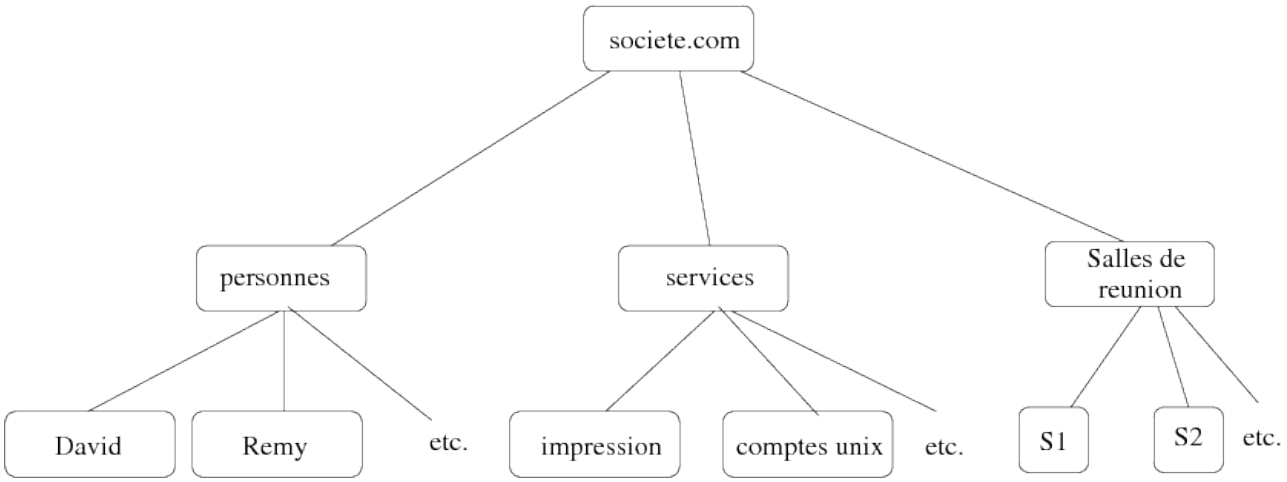


FIGURE 12.1: Exemple de Directory Information Tree (DIT)

Chaque noeud est une entrée de l'annuaire (Directory Service Entry, DSE).

Au sommet de l'arbre se trouve Root Entry (ou Suffix ou BaseDN) qui caractérise la base de données.

Les autres entrées appartiennent à différentes classes (`objectClass`) qui représentent différentes sortes d'objets (des groupes (objectClass `organizationalUnit`, ou), ou des personnes (objectClass `person` ou `inetOrgPerson`, uid), ou des sevices, etc.)

Chaque objectClass comporte des attributs qui caractérisent les différents objets de la classe. Il y a des atributs obligatoires (must) et des attributs facultatifs (may).

Par exemple, une personne de l'object class `person` doit avoir un `objectclass`, un nom de famille `sn`, nom usuel ou nom complet `cn`, et peut avoir un prénom (`givenname`), une `description`, un `telephonenumber`, un `userpassword`,...

Les classes peuvent dériver les unes des autres comme `inetOrgPerson` qui hérite des attributs de `person`.

Lorsqu'on crée une nouvelle `objectClass`, il faut la soumettre à l'Internet Assigned Numbers Authority (IANA) pour que la classe soit officiellement référencée dans LDAP.

12.2 Le Distinguished Name (DN)

Le DN d'une entrée est le chemin dans l'arbre jusqu'à cette entrée à partir de la racine (voir la figure 12.2). Il caractérise l'entrée et permet un accès rapide à cette entrée et à ses attributs dans le DIT.

Exemple : `cn=Remy, ou=personnes, dc=societe, dc=com`

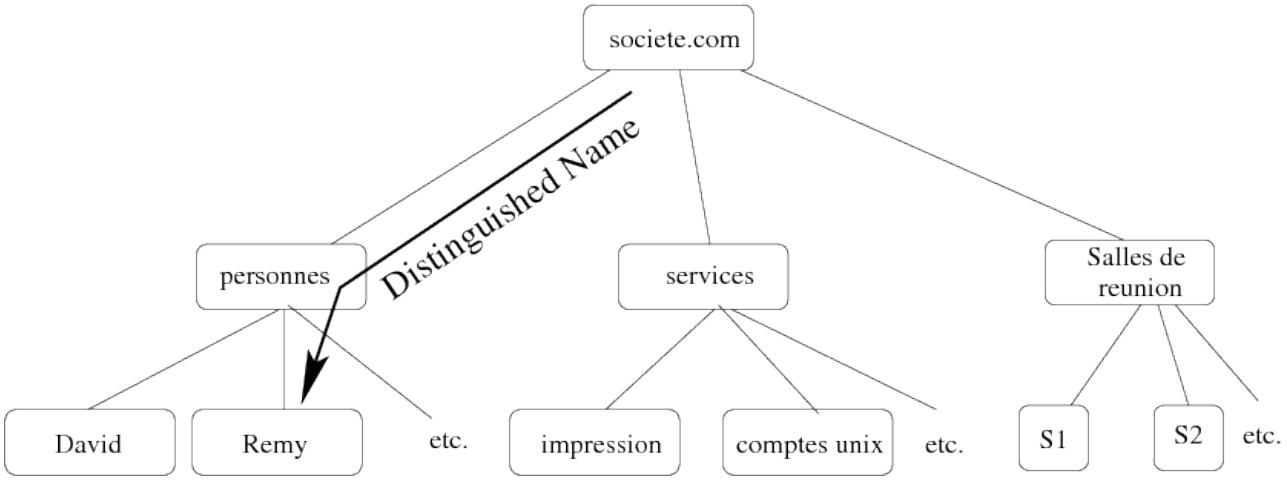


FIGURE 12.2: Exemple de Distinguished Name `dn:cn=Remy, ou=personnes, dc=societe, dc=com`.

12.3 Le modèle Fonctionnel

Le modèle fonctionnel perme d'effectuer des interrogations (recherches dans la base), de comparaison, de mises à jour (création, modification d'entrées), d'authentification (pour droits d'accès) et de contrôle sur le DIT.

12.3.1 Requêtes

Les requêtes permettent de rechercher des objets et d'afficher leurs attributs suivant différents critères.

- `base object` : l'endroit de l'arbre où doit commencer la recherche (voir figure 12.3). En effet, on n'effectue pas nécessairement la recherche dans l'arbre entier. Par exemple, si l'on recherche une personne, dans l'exemple de la figure 12.1, on recherchera en dessous de l'ou `personnes`. On évite ainsi le coût de la recherche dans le reste de l'arbre.

- `scope` : la profondeur de la recherche. Exemple : `scope = base` (1 seul noeud le base object), `scope=subtree` (tout le sous-arbre sous le base object).
- `derefAliases` : si on suit les liens vers d'autres BD ou pas
- `size limit` : nombre de réponses limite (garde-fou)
- `time limit` : temps maxi alloué pour la recherche (garde-fou)
- `attrOnly` : renvoie ou pas la valeur des attributs en plus de leur type
- `search filter` : le filtre de recherche (requête proprement dite).
- `list of attributes` : la liste des attributs que l'on souhaite connaître

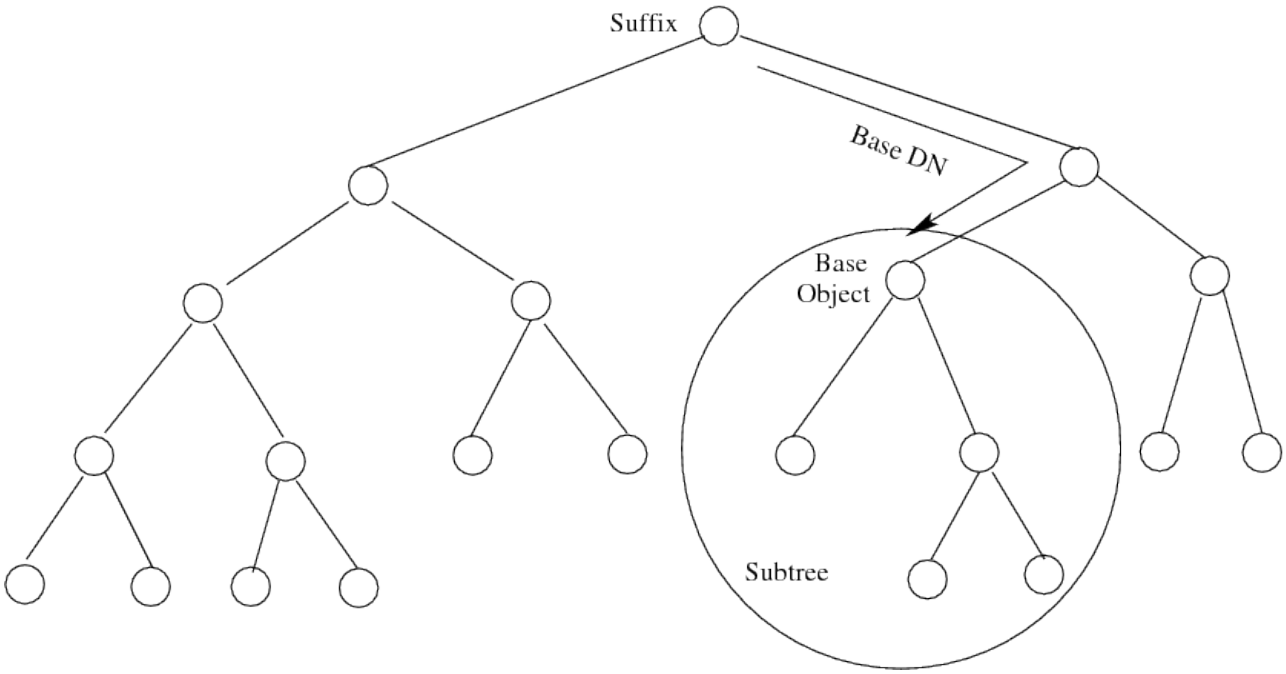


FIGURE 12.3: Le base object et le subtree d'une requête.

12.3.2 Les filtres de recherche

Syntaxe générale des filtres (¡operator¿(¡search operation¿)(¡search operation¿)...))
Exemples de filtres de recherche :

- `(sn=Rémy Malgouyres)` égalité Nom de famille vaut "Malgouyres"
- `(cn=*elon*)` sous-chaîne Nom contient "elon"
- `(cn~=Malgoire)`
  approximation le nom sonne comme "Malgoire"
- `(printedPages>=10000)` comparaison printedPages supérieur àă 10000

- `(sn=*)` existance Tous les noms propres
- `(&(ou=personnes)(telephoneNumber=*))` ET personne et l'attribut numéro de téléphone est connu
- `(|(ou=clermont)(ou=le-puy))` OU ou vaut clermont ou le-puy
- `(!(mail=*))` NON Toutes les entrées sans attribut "mail"

Exemple de filtre composé : `(&(objectclass=person)((tel=0473*)))` Toutes les entrées de type utilisateur avec numéro de téléphone commençant par 0473.

12.3.3 Le client `ldapsearch`

Le client `ldapsearch` (paquet debian `ldap-utils`) permet d'effectuer des requêtes en ligne de commande dans un annuaire LDAP.

On spécifie les caractéristiques de la requêtes (section 12.3.1) avec les options suivantes (liste non exhaustive, voir `man ldapsearch(1)` pour plus de détails) :

- `-b 'base object dn'` : spécification du base object (endroit où commencer la recherche dans l'arbre) ;
- `-H ldaphost : spécification du serveur LDAP à consulter` ;
- 'filtre' : filtre à appliquer pour la requête (voir section 12.3.2).
- atributes : les attributs des objets à afficher en sortie de la requête.
- `-D 'DN de l'utilisateur'` : connection en s'identifiant en tant qu'utilisateur via son DN.
- `-W` : prompter pour demander le mot de passe (s'utilise avec -D)
- `-x` : authentification simple non cryptée. Pour une connection plus sûre, utiliser par exemple SASL.

Exemple de requêtes effectuées en ligne de commande avec `ldapsearch` :
Affichage des personnes avec un numéro de téléphone sur le server LDAP `example.com` :

```
ldapsearch -x -H ldap://example.com -b 'dc=personnes,dc=societe,dc=com' 'telephoneNumb
```

Affichage du nom commun (cn) et de la description de tous les objets de la classe `simpleSecurityObject`
Connection en tant qu'administrateur avec prompt de mot de passe.

```
ldapsearch -x -D 'cn=admin,dc=societe,dc=com' -W -H ldap://bigboss -b 'dc=societe, dc=c
```

12.3.4 Mise à jour de la base

Les opérations de mise à jour permettent de modifier l'état de la base.

Opérations `add, delete, rename, modify` ;

- `add` : Ajout d'une entrée qui n'existe pas. Le parent de l'entrée doit avoir été préalablement créé. Les attributs objigatoires de l'entrée (suivant son objectClass) doivent être spécifiés.
- `rename` : Modification du DN d'une entrée. La nouvelle entrée doit avoir un parent existant. Tout le sous-arbre de cette entrée est déplacé.
- `modify` : modification du contenu (attributs, objectClass...) d'une entrée.
- `delete` : suppression d'une entrée. L'entrée ne doit pas avoir d'enfants. Supprimer les sous-arbres récursivement.

12.4 LDIF : LDAP Data Interchange Format

Format de fichier pour faire des imports/export d'entrées dans une base ou de bases ou bien faire des modifications sur les entrées.

Les fichiers LDIF sont codés en ASCII (norme UTF-8). Toute valeur qui n'est pas en ASCII (nombre, etc..) est codée en ASCII en base 64.

Pour importer les entrées d'un fichier LDIF en s'identifiant à partir d'un DN de la base (ici le DN "cn=admin, dc=societe, dc=com")

```
ldapadd -x -D 'cn=admin, dc=societe, dc=com' -f mon_fichier.ldif -W
```

La forme générale est :

```
dn: <distinguished name
objectClass: <object class>
objectClass: <object class>
[...]
attribute type:<attribute value>
attribute type:<attribute value>
[...]
```

Exemple de création d'Organisational Unit :

```
dn: ou=personnes,dc=societe,dc=com
objectClass: organizationalUnit
ou: personnes
```

Exemple d'entrée de type personne (voir figure 12.1) :

```
dn: cn=Rémy Malgouyres, ou=personnes, dc=societe, dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: remalgou
```

```
cn: Rémy Malgouyres
sn: Malgouyres
givenName: Rémy
mail: remy.malgouyres@societe.com
telephoneNumber: 04 73 17 70 00
userPassword: {sha}GBKC57D5LE

dn: cn=David Delon, ou=personnes, dc=societe, dc=com
objectClass: top
objectClass: person
objectClass: organizationalPersonobjectClass: inetOrgPerson
uid: dadelon
cn: David Delon
sn: Delon
givenName: David
mail: david.delon@societe.com
userPassword: {sha}FDHS5J34AH
```

La forme générale d'une mise à jour en mode commande est :

```
dn: distinguished name
changetype identifier
change operation identifier
list of attributes...
-
change operation identifier
list of attributes
```

Exemple :

```
dn: cn=Malgouyres, ou=personnes, dc=societe, dc=com
changetype: modify
replace: telephonenumber
telephonenumber: 04 73 17 70 00
-
add: mobile
mobilenumber = 06 07 08 09 10
```

12.5 Accès à la base à partir d'un client internet

syntaxe :

```
ldap[s]://<hostname>:<port>/<base_dn>?<attributes>?<scope>?<filter>
```

exemples :

Toute le sous arbre de ou=personnes,dc=societe,dc=com :

```
ldap://ldap.societe.com/ou=personnes,dc=societe,dc=com
```

Telephone number de toutes les personnes de nom de faille “Malgouyres”

```
ldap://ldap.societe.com/sn=Malgouyres,ou=personnes,dc=societe,dc=com?telephonenumber
```

Adresses mail de tous les utilisateurs de gmail

```
ldap://ldap.societe.com/ou=personnes,dc=societe,dc=com?mail?subtree?mail=*@gmail.com
```

12.6 Administrer OpenLDAP : `slapd.conf`

Le premier fichier de configuration d'OpenLDAP est "/etc/ldap/sldapd.conf" qui décrit les principaux paramètres de votre annuaire (voir man slapd.conf pour les nombreuses options) :

```
# $OpenLDAP: pkg/ldap/servers/slapd/slapd.conf,v 1.8.8.6 2001/04/20 23:32:43 kurt Exp $
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
# Inclusion des schémas nécessaires
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema

# Options que vous pouvez modifier
#pidfile        //var/run/slapd.pid
#argsfile       //var/run/slapd.args

########################################################################
# ldbm database definitions
########################################################################
# Choix du format de base de données pour le stockage des informations.
database        ldbm

# Configurer le suffixe (racine) de l'annuaire
# en fonction du domaine  DNS
suffix          "dc=societe,dc=com"
# ou d'une autre organisation
#suffix         "o=Ma Societe,c=FR"

# L'administrateur de l'annuaire
rootdn          "cn=Manager,dc=societe,dc=com"

# Le mot de passe de l'administrateur, préférer une option cryptée
# La commande htpasswd peut très bien faire l'affaire pour encrypter

# rootpw                secret
# rootpw                {crypt}ijFYNcSNctBYg
```

```
# Emplacement de la base de données
directory       /var/lib/ldap

# Création des index.
# Comme pour une base de données, indexer les rubriques
# les plus utilisées.

index   objectClass,uid,uidNumber,gidNumber,memberUid   eq
index   cn,mail,surname,givenname                       eq,subinitial

# La réplication ne sera  pas utilisée ici
# Vous pouvez répliquer tout ou partie d'un arbre
# activée par le daemon slurpd

# Directives de replication
# sinon les mettre dans un fichier à part et utiliser
# replogfile /chemincomplet/du/fichier

# Indiquer quels sont les serveurs réplicats
# et la méthode d'authentification
# Ici le serveur local, se répliquera sur ldap1

#replica host:ldap-1.example.com:389
#       bindmethod=simple
#       binddn="cn=replicat_slave1, dc=mydomain, dc=fr"
#       credential=UnMotDePasse

# Accés par défaut sur la base
defaultaccess read
```

Pour modifier les droits d'accès :

```
access to <what> [ by <who> <none | compare | search | read | write>]

# Donne un accés en écriture pour le manager du domaine
access to * by dn="cn=Manager,dc=mydomain,dc=fr" write
# Donne un accés en lecture à tout le monde sur la base
access to * by * read
# Donne un accés en écriture sur un attribut pour le manager
#                en lecture pour les autres.
access to attr=uid
                 by dn="manager,dc=mydomain,dc=fr" write
                 by * none
```