Dernière mise-à-jour : 2020/08/07 17:20

LDF306 - Gestion du Réseau

Contenu du Module

- Diagnostique du Réseau
  - ping
  - netstat -i
  - traceroute
- Connexions à Distance
  - Telnet
  - wget
  - ftp
  - SSH
    - Introduction
      - SSH-1
      - SSH-2
    - L'authentification par mot de passe
    - L'authentification par clef asymétrique
      - Installation
      - Configuration
        - Serveur
      - Utilisation
      - Tunnels SSH
  - SCP
    - Introduction
    - Utilisation
    - Mise en place des clefs
- Le Pare-feu Netfilter
  - Configuration du Pare-feu Netfilter/iptables
    - Introduction
    - Configuration par Scripts sous Debian 6 et 7
  - La Configuration par firewalld sous Debian 8
    - La Configuration de Base de firewalld
    - La Commande firewall-cmd
    - La Configuration Avancée de firewalld
    - Le mode Panic de firewalld
- Annexe #1 - Comprendre les Réseaux
  - Présentation des Réseaux

- Classification des Réseaux
    - Classification par Mode de Transmission
    - Classification par Topologie
        - La Topologie Physique
        - La Topologie en Ligne
        - La Topologie en Bus
        - La Topologie en Étoile
        - La Topologie en Anneau
        - La Topologie en Arbre
        - La Topologie Maillée
    - Classification par Étendue
    - Les Types de LAN
        - Réseau à Serveur Dédié
        - Réseau Poste-à-Poste
- Le Modèle Client/Serveur
- Modèles de Communication
    - Le modèle OSI
        - Les Couches
        - Les Protocoles
        - Les Interfaces
        - Protocol Data Units
        - Encapsulation et Désencapsulation
    - Spécification NDIS et le Modèle ODI
    - Le modèle TCP/IP
- Les Raccordements
    - Les Modes de Transmission
    - Les Câbles
        - Le Câble Coaxial
        - Le Câble Paire Torsadée
        - Catégories de Blindage
        - La Prise RJ45
        - Channel Link et Basic Link
        - La Fibre Optique
    - Les Réseaux sans Fils

- Fonctions de Hachage
- Signature Numérique
- LAB #1 - Utilisation de GnuPG
    - Présentation
    - Installation
    - Configuration
    - Signer un message
    - Chiffrer un message
- PKI
    - Certificats X509

Configuration de TCP/IP sous Debian 6

La configuration TCP/IP se trouve dans le fichier **/etc/network/interfaces** :

DHCP

/etc/network/interfaces

```
root@debian6:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
```

```
#NetworkManager#iface eth0 inet dhcp
```

Dans ce fichier chaque déclaration est de la forme suivante :

```
interface   nom    type    mode
```

On peut constater donc dans notre exemple ci-dessus :

- une déclaration pour l'interface **lo** de loopback
- une déclaration pour l'interface **eth0** en dhcp

IP Fixe

Dans le cas où l'interface eth0 était configuré en IP statique, la déclaration concernant eth0 prendrait la forme suivante :

```
auto eth0
iface eth0 inet static
        address 10.0.2.15
        netmask 255.255.255.0
        broadcast 10.0.2.255
        network 10.0.2.0
        gateway 10.0.2.2
```

Dans ce fichier vous pouvez constater les directives suivantes :

Directive	**Description**
address	Indique l'adresse IPv4 de l'interface
netmask	Indique le masque de sous-réseau IPv4
broadcast	Indique l'adresse de diffusion IPv4
network	Indique l'adresse réseau IPv4
gateway	Indique l'adresse IPv4 de la passerelle par défaut

Notez que VirtualBox fournit une passerelle par défaut (10.0.2.2).

Après avoir modifier le fichier **/etc/network/interfaces** vous devez arrêter le service **network-manager** utilisé pour la connexion DHCP et activer le service **networking** :

```
root@debian6:~# service network-manager stop
Stopping network connection manager: NetworkManager.
root@debian6:~# update-rc.d -f network-manager remove
update-rc.d: using dependency based boot sequencing
root@debian6:~# chkconfig --level 2345 networking on
root@debian6:~# service networking start
Configuring network interfaces...done.
```

Si le service networking réfuse de démarrer en produisant une erreur, le problème vient certainement du fait que votre interface réseau a été configurée par **udev** en **eth1**. La solution la plus simple est d'éditer le fichier **/etc/udev/rules.d/70-persistent-net.rules** en supprimant toutes les lignes qui ne commencent pas par le caractère **#** et de re-démarrer votre machine virtuelle.

La Commande hostname

Le nom de la machine se trouve dans le fichier **/etc/hostname** :

```
root@debian6:~# cat /etc/hostname
debian6
```

Ce nom doit être un FQDN (*Fully Qualified Domain Name*). Modifiez donc ce fichier ainsi :

```
root@debian6:~# cat /etc/hostname
```

```
debian6.fenestros.loc
```

Afin d'informer le système immédiatement de la modification du FQDN, utilisez la commande **hostname** :

```
root@debian6:~# hostname
debian6
root@debian6:~# hostname debian6.fenestros.loc
root@debian6:~# hostname
debian6.fenestros.loc
```

Pour afficher le FQDN du système vous pouvez également utiliser la commande suivante :

```
root@debian6:~# uname -n
debian6.fenestros.loc
```

Options de la commande hostname

Les options de cette commande sont :

```
root@debian6:~# hostname --help
Usage: hostname [-v] [-b] {hostname|-F file}         set host name (from file)
       hostname [-v] [-d|-f|-s|-a|-i|-y|-A|-I]             display formatted name
       hostname [-v]                                 display host name

       {yp,nis,}domainname [-v] {nisdomain|-F file}  set NIS domain name (from file)
       {yp,nis,}domainname [-v]                      display NIS domain name

       dnsdomainname [-v]                            display dns domain name

       hostname -V|--version|-h|--help               print info and exit

Program name:
```

```
       {yp,nis,}domainname=hostname -y
       dnsdomainname=hostname -d

Program options:
    -s, --short            short host name
    -a, --alias            alias names
    -i, --ip-address       addresses for the host name
    -I, --all-ip-addresses all addresses for the host
    -f, --fqdn, --long     long host name (FQDN)
    -A, --all-fqdns        all long host names (FQDNs)
    -d, --domain           DNS domain name
    -y, --yp, --nis        NIS/YP domain name
    -b, --boot             set default hostname if none available
    -F, --file             read host name or NIS domain name from given file

Description:
   This command can get or set the host name or the NIS domain name. You can
   also get the DNS domain or the FQDN (fully qualified domain name).
   Unless you are using bind or NIS for host lookups you can change the
   FQDN (Fully Qualified Domain Name) and the DNS domain name (which is
   part of the FQDN) in the /etc/hosts file.
```

La Commande ifconfig

Pour afficher la configuration IP de la machine il faut saisir la commande suivante :

```
root@debian6:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:2a:02:5c
          inet adr:10.0.2.15  Bcast:10.0.2.255  Masque:255.255.255.0
          adr inet6: fe80::a00:27ff:fe2a:25c/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:990 errors:0 dropped:0 overruns:0 frame:0
          TX packets:580 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 lg file transmission:1000
          RX bytes:684107 (668.0 KiB)  TX bytes:97392 (95.1 KiB)

lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:560 (560.0 B)  TX bytes:560 (560.0 B)
```

La commande ifconfig est également utilisée pour configurer une interface.

Créez maintenant une interface fictive ainsi :

```
root@debian6:~# ifconfig eth0:1 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255
```

Constatez maintenant le résultat :

```
root@debian:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:2a:02:5c
          inet adr:10.0.2.15  Bcast:10.0.2.255  Masque:255.255.255.0
          adr inet6: fe80::a00:27ff:fe2a:25c/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1013 errors:0 dropped:0 overruns:0 frame:0
          TX packets:611 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:686171 (670.0 KiB)  TX bytes:100060 (97.7 KiB)

eth0:1    Link encap:Ethernet  HWaddr 08:00:27:2a:02:5c
          inet adr:192.168.1.2  Bcast:192.168.1.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:560 (560.0 B)  TX bytes:560 (560.0 B)
```

Options de la commande ifconfig

Les options de cette commande sont :

```
root@debian:~# ifconfig --help
Usage:
  ifconfig [-a] [-v] [-s] <interface> [[<AF>] <address>]
  [add <adresse>[/<lg_prefixe>]]
  [del <adresse>[/<lg_prefixe>]]
  [[-]broadcast [<adresse>]]  [[-]pointopoint [<adresse>]]
  [netmask <address>]  [dstaddr <address>]  [tunnel <address>]
  [outfill <NN>] [keepalive <NN>]
  [hw <HW> <adresse>]  [metric <NN>]  [mtu <NN>]
  [[-]trailers]  [[-]arp]  [[-]allmulti]
  [multicast]  [[-]promisc]
  [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <type>]
  [txqueuelen <NN>]
  [[-]dynamic]
  [up|down] ...

  <HW>=Type de matériel.
  Liste des types de matériels possibles:
    loop (Boucle locale) slip (IP ligne série) cslip (IP ligne série - VJ )
    slip6 (IP ligne série - 6 bits) cslip6 (IP ligne série - 6 bits VJ) adaptive (IP ligne série adaptative)
```

```
    strip (Metricom Starmode IP) ash (Ash) ether (Ethernet)
    tr (16/4 Mbps Token Ring) tr (16/4 Mbps Token Ring (New)) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) rose (AMPR ROSE) tunnel (IPIP Tunnel)
    ppp (Protocole Point-à-Point) hdlc ((Cisco)-HDLC) lapb (LAPB)
    arcnet (ARCnet) dlci (Frame Relay DLCI) frad (Périphériue d'accès Frame Relay)
    sit (IPv6-dans-IPv4) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
    irda (IrLAP) ec (Econet) x25 (generic X.25)
    eui64 (Generic EUI-64)
  <AF>=famille d'Adresses. Défaut: inet
  Liste des familles d'adresses possibles:
    unix (Domaine UNIX) inet (DARPA Internet) inet6 (IPv6)
    ax25 (AMPR AX.25) netrom (AMPR NET/ROM) rose (AMPR ROSE)
    ipx (Novell IPX) ddp (Appletalk DDP) ec (Econet)
    ash (Ash) x25 (CCITT X.25)
```

Activer/Désactiver une Interface Manuellement

Deux commandes existent pour activer et désactiver manuellement une interface réseau :

```
root@debian6:~# ifdown eth0
root@debian6:~# ifconfig
lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:560 (560.0 B)  TX bytes:560 (560.0 B)

root@debian6:~# ifup eth0
root@debian6:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:2a:02:5c
```

```
          inet adr:10.0.2.15  Bcast:10.0.2.255  Masque:255.255.255.0
          adr inet6: fe80::a00:27ff:fe2a:25c/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2556 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1632 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:2893516 (2.7 MiB)  TX bytes:176291 (172.1 KiB)

eth0:1    Link encap:Ethernet  HWaddr 08:00:27:2a:02:5c
          inet adr:192.168.1.2  Bcast:192.168.1.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:560 (560.0 B)  TX bytes:560 (560.0 B)
```

/etc/networks

Ce fichier contient la correspondance entre des noms de réseaux et l'adresse IP du réseau :

```
root@debian6:~# cat /etc/networks
default     0.0.0.0
loopback    127.0.0.0
link-local  169.254.0.0
```

Résolution d'adresses IP

La configuration DNS est stockée dans le fichier **/etc/resolv.conf**.

/etc/resolv.conf

La configuration DNS est stockée dans le fichier /etc/resolv.conf :

```
root@debian:~# cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Notez que les DNS utilisés sont les serveurs DNS publics de Google.

/etc/nsswitch.conf

L'ordre de recherche des services de noms est stocké dans le fichier **/etc/nsswitch.conf**. Pour connaître l'ordre, saisissez la commande suivante :

```
root@debian6:~# grep '^hosts:' /etc/nsswitch.conf
hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
```

/etc/hosts

Le mot **files** dans la sortie de la commande précédente fait référence au fichier **/etc/hosts** :

```
root@debian6:~# cat /etc/hosts
```

```
127.0.0.1   localhost
127.0.1.1   debian6.fenestros.loc    debian6

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Pour tester le serveur DNS, deux commandes sont possibles :

```
root@debian6:~# nslookup www.linuxelearning.com
Server:     8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
www.linuxelearning.com  canonical name = linuxelearning.com.
Name:   linuxelearning.com
Address: 212.198.31.61

root@debian6:~# dig www.linuxelearning.com

; <<>> DiG 9.7.3 <<>> www.linuxelearning.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45521
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.linuxelearning.com.        IN    A

;; ANSWER SECTION:
www.linuxelearning.com. 42847    IN    CNAME    linuxelearning.com.
```

```
linuxelearning.com. 60    IN    A    212.198.31.61

;; Query time: 51 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed May  9 15:47:18 2012
;; MSG SIZE  rcvd: 70
```

Services réseaux

Quand un client émet une demande de connexion vers une application réseau sur un serveur, il utilise un socket attaché à un port local **supérieur à 1023**, alloué d'une manière dynamique. La requête contient le port de destination sur le serveur. Certaines applications serveurs se gèrent toutes seules, ce qui est la cas par exemple d'**httpd**. Par contre d'autres sont gérées par le service **xinetd**.

xinetd

Sous Debian 6 xinetd n'est pas installé par défaut. Installez-le grâce à apt-get :

```
root@debian6:~# apt-get install xinetd
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les NOUVEAUX paquets suivants seront installés :
  xinetd
0 mis à jour, 1 nouvellement installés, 0 à enlever et 219 non mis à jour.
Il est nécessaire de prendre 136 ko dans les archives.
Après cette opération, 311 ko d'espace disque supplémentaires seront utilisés.
Réception de : 1 http://ftp.fr.debian.org/debian/ squeeze/main xinetd i386 1:2.3.14-7 [136 kB]
136 ko réceptionnés en 0s (427 ko/s)
Sélection du paquet xinetd précédemment désélectionné.
(Lecture de la base de données... 130628 fichiers et répertoires déjà installés.)
Dépaquetage de xinetd (à partir de .../xinetd_1%3a2.3.14-7_i386.deb) ...
```

```
Traitement des actions différées (« triggers ») pour « man-db »...
Paramétrage de xinetd (1:2.3.14-7) ...
Stopping internet superserver: xinetd.
Starting internet superserver: xinetd.
```

Le programme xinetd est configuré via le fichier **/etc/xinetd.conf** :

```
root@debian6:~# cat /etc/xinetd.conf
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{

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

}

includedir /etc/xinetd.d
```

Les valeurs des directives dans le fichier **/etc/xinetd.conf** sont héritées par toutes les configurations des services sauf dans le cas où une variable est explicitement fixée dans un des fichiers de définitions des services se trouvant dans **/etc/xinetd.d**.

Les variables les plus usuellement utilisées dans **/etc/xinetd.conf** sont :

Directive	Déscription
instances	Le nombre de demandes d'accès simultanés
log_type	Indique à xinetd d'adresser les traces à SYSLOG avec l'étiquette de sous-système applicatif **daemon** et la priorité de **info**
log_on_succes	Indique que SYSLOG doit journaliser le PID, HOST, DURATION et EXIT en cas de succès
log_on_failure	Indique que SYSLOG doit journaliser le HOST en cas d'échec

Directive	Déscription
cps	Indique 50 connexions par seconde avec un temps d'indisponibilité de 10 secondes si le seuil est atteint

Les options concernant les journaux sont :

HOST	Journalisation de l'adresse IP de l'hôte distant
PID	Journalisation du PID du processus qui reçoit la demande d'accès
DURATION	Journalisation des durées d'utilisation
EXIT	Journalisation de l'état ou di signal de fin de service
ATTEMPT	Journalisation des connexions en échec
USERID	Journalisation des données concernant l'utilisateur selon la RFC 1413

Examinons maintenant le répertoire **/etc/xinetd.d** :

```
root@debian6:~# ls -l /etc/xinetd.d
total 20
-rw-r--r-- 1 root root 798 26 mars   2008 chargen
-rw-r--r-- 1 root root 660 26 mars   2008 daytime
-rw-r--r-- 1 root root 549 26 mars   2008 discard
-rw-r--r-- 1 root root 580 26 mars   2008 echo
-rw-r--r-- 1 root root 727 26 mars   2008 time
```

A l'examen de ce répertoire vous noterez que celui-ci contient des fichiers nominatifs par application-serveur, par exemple pour le serveur chargen :

```
root@debian6:~# cat /etc/xinetd.d/chargen
# default: off
# description: An xinetd internal service which generate characters.  The
# xinetd internal service which continuously generates characters until the
# connection is dropped.  The characters look something like this:
# !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefg
# This is the tcp version.
service chargen
{
```

```
    disable        = yes
    type        = INTERNAL
    id        = chargen-stream
    socket_type    = stream
    protocol    = tcp
    user        = root
    wait        = no
}

# This is the udp version.
service chargen
{
    disable        = yes
    type        = INTERNAL
    id        = chargen-dgram
    socket_type    = dgram
    protocol    = udp
    user        = root
    wait        = yes
}
```

Les directives principales de ce fichier sont :

Paramètre	Déscription
disable	**no** : Le service est actif. **yes** : Le service est désactivé
port	Le numéro de port ou, à défaut, le numéro indiqué pour le service dans le fichier /etc/services
socket_type	Nature du socket, soit **stream** pour TCP soit **dgram** pour UDP
protocol	Protocole utilisé soit **TCP** soit **UDP**
wait	**no** : indique si xinetd active un serveur par client. **yes** : indique que xinetd active un seul serveur pour tous les client
user	Indique le compte sous lequel le serveur est exécuté
server	Indique le chemin d'accès de l'application serveur
env	Définit un environnement système
server_args	Donne les arguments transmis à l'application serveur

Cependant il est aussi possible d'utiliser les directives suivantes :

Paramètre	Déscription	Exemple
nice	Fixe le niveau de nice entre -19 et +20	10
max_load	Fixe la charge CPU maximum admise. Au delà aucune connexion supplémenatire en sera acceptée	2.5
bind	Limite le service à l'interface dont l'adresse IP est indiquée	192.168.1.1
only_from	Limite le service aux seuls clients indiqués par la plage donnée	192.168.1.0/24 fenestros.loc
no_access	Interdit le service aux clients indiqués par la plage donnée	192.168.2.0/24, i2tch.loc
access_time	Limite l'accès au service à une plage horaire	09:00-19:00
redirect	Redirige les requêtes sur un port donné à une autre adresse IP	192.168.1.10 23

Afin d'activer une application serveur, il suffit de modifier le paramètre **disable** dans le fichier concerné et de relancer le service xinetd.

TCP Wrapper

TCP Wrapper contrôle l'accès à des services réseaux grâce à des **ACL**.

Quand une requête arrive pour un serveur, xinetd active le wrapper **tcpd** au lieu d'activer le serveur directement.

tcpd met à jour un journal et vérifie si le client a le droit d'utiliser le service concerné. Les ACL se trouvent dans deux fichiers:

- **/etc/hosts.allow**
- **/etc/hosts.deny**

Il faut noter que si ces fichiers n'existent pas ou sont vides, il n'y a pas de contrôle d'accès.

Le format d'une ligne dans un de ces deux fichiers est:

```
démon : liste_de_clients
```

Par exemple dans le cas d'un serveur **démon**, on verrait une ligne dans le fichier **/etc/hosts.allow** similaire à:

```
démon : LOCAL, .fenestros.loc
```

ce qui implique que les machines dont le nom ne comporte pas de point ainsi que les machines du domaine **fenestros.loc** sont autorisées à utiliser le service.

Le mot clef **ALL** peut être utilisé pour indiquer tout. Par exemple, **ALL:ALL** dans le fichier **/etc/host.deny** bloque effectivement toute tentative de connexion à un service xinetd sauf pour les ACL inclus dans le fichier /etc/host.allow.

Routage Statique

La Commande route

Pour afficher la table de routage de la machine vous pouvez utiliser la commande **route** :

```
root@debian6:~# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
10.0.2.0        *               255.255.255.0   U     0      0        0 eth0
default         10.0.2.2        0.0.0.0         UG    0      0        0 eth0
```

La table issue de la commande **route** indique les informations suivantes:

- La destination qui peut être un hôte ou un réseau et est identifiée par les champs **Destination** et **Genmask**
- La route à prendre identifiée par les champs **Gateway** et **Iface**. Dans le cas d'une valeur de 0.0.0.0 ceci spécifie une route directe. La valeur d'Iface spécifie la carte à utiliser,
- Le champ **Indic** qui peux prendre un ou plusieurs de svaleurs suivantes:
  - U - **Up** - la route est active
  - H - **Host** - la route conduit à un hôte
  - G - **Gateways** - la route passe par une passerelle
- Le champ **Metric** indique le nombre de sauts (passerelles) pour atteindre la destination,
- Le champ **Ref** indique le nombre de références à cette route. Ce champs est usilisé par le Noyau de Linux,

- Le champ **Use** indique le nombre de recherches associés à cette route.

La commande **route** permet aussi de paramétrer le routage indirect. Par exemple pour supprimer la route vers le réseau 192.168.1.0 :

```
root@debian6:~# route del -net 192.168.1.0 netmask 255.255.255.0
root@debian6:~# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
10.0.2.0        *               255.255.255.0   U     0      0        0 eth0
default         10.0.2.2        0.0.0.0         UG    0      0        0 eth0
```

Pour ajouter la route vers le réseau 192.168.1.0 :

```
root@debian6:~# route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.1.2
root@debian6:~# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     mdt34.formation 255.255.255.0   UG    0      0        0 eth0
10.0.2.0        *               255.255.255.0   U     0      0        0 eth0
default         10.0.2.2        0.0.0.0         UG    0      0        0 eth0
```

La commande utilisée pour ajouter une passerelle par défaut prend la forme suivante **route add default gw *numéro_ip interface***.

Les options cette commande sont :

```
root@debian6:~# route --help
Syntaxe: route [-nNvee] [-FC] [<AF>]           Liste les tables de routage noyau
       route [-v] [-FC] {add|del|flush} ...  Modifie la table de routage pour AF.

       route {-h|--help} [<AF>]              Utilisation détaillée pour l'AF spécifié.
       route {-V|--version}                  Affiche la version/auteur et termine.
```

```
       -v, --verbose            mode verbeux
       -n, --numeric            don't resolve names
       -e, --extend             display other/more information
       -F, --fib                display Forwarding Information Base (default)
       -C, --cache              affiche le cache de routage au lieu de FIB

  <AF>=Use '-A <af>' or '--<af>'; default: inet
  Liste les familles d'adresses possibles (supportant le routage):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
    x25 (CCITT X.25)
```

Vous pouvez aussi utiliser la commande **netstat** pour afficher la table de routage de la machine :

```
root@debian6:~# netstat -nr
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic   MSS Fenêtre irtt Iface
192.168.1.0     192.168.1.2     255.255.255.0   UG        0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         10.0.2.2        0.0.0.0         UG        0 0          0 eth0
```

La table issue de la commande **netstat -nr** indique les informations suivantes:

- La champ **MSS** indique la taille maximale des segments TCP sur la route,
- Le champ **Window** indique la taille de la fenêtre sur cette route,
- Le champ **irrt** indique le paramètre IRRT pour la route.

Activer/désactiver le routage sur le serveur

Pour activer le routage sur le serveur, il convient d'activer la retransmission des paquets:

```
root@debian6:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@debian6:~# cat /proc/sys/net/ipv4/ip_forward
```

```
1
```

Pour désactiver le routage sur le serveur, il convient de désactiver la retransmission des paquets:

```
root@debian6:~# echo 0 > /proc/sys/net/ipv4/ip_forward
root@debian6:~# cat /proc/sys/net/ipv4/ip_forward
0
```

Configuration du Réseau sous Debian 8

Debian 8 utilise exclusivement **Network Manager** pour gérer le réseau. Network Manager est composé de deux éléments :

- un service qui gère les connexions réseaux et rapporte leurs états,
- des front-ends qui passent par un API de configuration du service.

> **Important** : Notez qu'avec cette version de NetworkManager, IPv6 est activée par défaut.

Le service NetworkManager doit toujours être lancé :

```
root@debian8:~# systemctl status NetworkManager.service
● NetworkManager.service - Network Manager
   Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled)
   Active: active (running) since Mon 2015-11-02 14:01:23 CET; 20h ago
 Main PID: 510 (NetworkManager)
   CGroup: /system.slice/NetworkManager.service
           ├─ 510 /usr/sbin/NetworkManager --no-daemon
           └─1715 /sbin/dhclient -d -q -sf /usr/lib/NetworkManager/nm-dhcp-helper -pf /var/run/dhclient-eth0.pid
-lf /var/lib/NetworkManager/dhclie...

Nov 03 09:10:12 debian8 NetworkManager[510]: <info> Activation (eth0) Stage 5 of 5 (IPv4 Commit) started...
```

```
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> (eth0): device state change: ip-config -> ip-check (reason
'none') [70 80 0]
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> Activation (eth0) Stage 5 of 5 (IPv4 Commit) complete.
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> (eth0): device state change: ip-check -> secondaries (reason
'none') [80 90 0]
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> (eth0): device state change: secondaries -> activated (reason
'none') [90 100 0]
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> NetworkManager state is now CONNECTED_LOCAL
Nov 03 09:10:12 debian8 dhclient[1715]: bound to 10.0.2.15 -- renewal in 42103 seconds.
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> NetworkManager state is now CONNECTED_GLOBAL
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> Policy set 'Wired connection 1' (eth0) as default for IPv4
routing and DNS.
Nov 03 09:10:12 debian8 NetworkManager[510]: <info> Activation (eth0) successful, device activated.
```

La Commande nmcli

La commande **nmcli** (Network Manager Command Line Interface) est utilisée pour configurer NetworkManager.

Les options et les sous-commandes peuvent être consultées en utilisant les commandes suivantes :

```
root@debian8:~# nmcli --help
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }

OPTIONS
  -t[erse]                                  terse output
  -p[retty]                                 pretty output
  -m[ode] tabular|multiline                 output mode
  -f[ields] <field1,field2,...>|all|common   specify fields to output
  -e[scape] yes|no                          escape columns separators in values
  -n[ocheck]                                don't check nmcli and NetworkManager versions
  -a[sk]                                    ask for missing parameters
  -w[ait] <seconds>                         set timeout waiting for finishing operations
  -v[ersion]                                show program version
```

```
  -h[elp]                                  print this help

OBJECT
  g[eneral]       NetworkManager's general status and operations
  n[etworking]    overall networking control
  r[adio]         NetworkManager radio switches
  c[onnection]    NetworkManager's connections
  d[evice]        devices managed by NetworkManager

root@debian8:~# nmcli g help
Usage: nmcli general { COMMAND | help }

COMMAND := { status | hostname | permissions | logging }

  status

  hostname [<hostname>]

  permissions

  logging [level <log level>] [domains <log domains>]

root@debian8:~# nmcli g status help
Usage: nmcli general status { help }

Show overall status of NetworkManager.
'status' is the default action, which means 'nmcli gen' calls 'nmcli gen status'
```

Connections et Profils

NetworkManager inclus la notion de **connections** ou **profils** permettant des configurations différentes en fonction de la localisation. Pour voir les connections actuelles, utilisez la commande **nmcli c** avec la sous-commande **show** :

```
root@debian8:~# nmcli c show
NAME                UUID                                  TYPE            DEVICE
Wired connection 1  0caa0d4f-0222-415a-8a4f-753ca284911e  802-3-ethernet  eth0
```

Comme on peut constater ici, il n'existe pour le moment, qu'un seul profil.

Si vous constatez que vous n'avez pas de profil dans la sortie de la commande **nmcli c show**, ceci indique que NetworkManager ne gère pas la carte :

```
root@debian8:~# nmcli c show
NAME     UUID      TYPE      DEVICE
```

Dans ce cas, le fichier **/etc/network/interfaces** contiendra une entrée pour la carte **eth0** :

```
root@debian8:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp
```

Afin de permettre NetworkManager de gérer la carte malgré cette configuration, il convient d'éditer le fichier **/etc/NetworkManager/NetworkManager.conf** et passer la directive **managed=false** à **managed=true** :

```
root@debian8:~# cat /etc/NetworkManager/NetworkManager.conf
[main]
plugins=ifupdown,keyfile
```

```
[ifupdown]
managed=true
```

Il faut ensuite re-démarrer le service NetworkManager :

```
root@debian8:~# systemctl restart NetworkManager.service
```

Créez maintenant un profil IP fixe rattaché au périphérique **eth0** :

```
root@debian8:~# nmcli connection add con-name ip_fixe ifname eth0 type ethernet ip4 10.0.2.16/24 gw4 10.0.2.2
Connection 'ip_fixe' (ec7e2b5f-0d19-4a40-8639-24af88a59d79) successfully added.
```

Constatez sa présence :

```
root@debian8:~# nmcli c show
NAME                UUID                                  TYPE            DEVICE
ip_fixe             ec7e2b5f-0d19-4a40-8639-24af88a59d79  802-3-ethernet  --
Wired connection 1  0caa0d4f-0222-415a-8a4f-753ca284911e  802-3-ethernet  eth0
```

Notez que la sortie n'indique pas que le profil **ip_fixe** soit associé au periphérique **enp0s3** car le profil **ip_fixe** n'est pas activé :

```
root@debian8:~# nmcli d show
GENERAL.DEVICE:                         eth0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         08:00:27:F2:8C:8F
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     Wired connection 1
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/2
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         ip = 10.0.2.15/24, gw = 10.0.2.2
IP4.ROUTE[1]:                           dst = 169.254.0.0/16, nh = 0.0.0.0, mt = 1000
IP4.DNS[1]:                             8.8.8.8
IP4.DOMAIN[1]:                          formation.local
```

```
IP6.ADDRESS[1]:                         ip = fe80::a00:27ff:fef2:8c8f/64, gw = ::

GENERAL.DEVICE:                         lo
GENERAL.TYPE:                           loopback
GENERAL.HWADDR:                         00:00:00:00:00:00
GENERAL.MTU:                            65536
GENERAL.STATE:                          10 (unmanaged)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
IP4.ADDRESS[1]:                         ip = 127.0.0.1/8, gw = 0.0.0.0
IP6.ADDRESS[1]:                         ip = ::1/128, gw = ::
```

Pour activer le profil ip_fixe, utilisez la commande suivante :

```
root@debian8:~# nmcli connection up ip_fixe
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
```

Le profil ip_fixe est maintenant activé tandis que le profil Wired connection 1 a été désactivé :

```
root@debian8:~# nmcli c show
NAME                UUID                                  TYPE            DEVICE
ip_fixe             ec7e2b5f-0d19-4a40-8639-24af88a59d79  802-3-ethernet  eth0
Wired connection 1  0caa0d4f-0222-415a-8a4f-753ca284911e  802-3-ethernet  --
root@debian8:~# nmcli d show
GENERAL.DEVICE:                         eth0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         08:00:27:F2:8C:8F
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     ip_fixe
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/3
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         ip = 10.0.2.16/24, gw = 10.0.2.2
IP4.ROUTE[1]:                           dst = 169.254.0.0/16, nh = 0.0.0.0, mt = 1000
```

```
IP6.ADDRESS[1]:                         ip = fe80::a00:27ff:fef2:8c8f/64, gw = ::

GENERAL.DEVICE:                         lo
GENERAL.TYPE:                           loopback
GENERAL.HWADDR:                         00:00:00:00:00:00
GENERAL.MTU:                            65536
GENERAL.STATE:                          10 (unmanaged)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
IP4.ADDRESS[1]:                         ip = 127.0.0.1/8, gw = 0.0.0.0
IP6.ADDRESS[1]:                         ip = ::1/128, gw = ::
```

Pour consulter les paramètres d'un profil, utilisez la commande suivante :

```
root@debian8:~# nmcli -p connection show "Wired connection 1"
===============================================================================
                Connection profile details (Wired connection 1)
===============================================================================
connection.id:                          Wired connection 1
connection.uuid:                        0caa0d4f-0222-415a-8a4f-753ca284911e
connection.interface-name:              --
connection.type:                        802-3-ethernet
connection.autoconnect:                 yes
connection.timestamp:                   1446544245
connection.read-only:                   no
connection.permissions:
connection.zone:                        --
connection.master:                      --
connection.slave-type:                  --
connection.secondaries:
connection.gateway-ping-timeout:        0
-------------------------------------------------------------------------------
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
```

```
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          yes
802-3-ethernet.mac-address:             --
802-3-ethernet.cloned-mac-address:      --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:                     auto
802-3-ethernet.s390-subchannels:
802-3-ethernet.s390-nettype:            --
802-3-ethernet.s390-options:
---------------------------------------------------------------------------
ipv4.method:                            auto
ipv4.dns:
ipv4.dns-search:
ipv4.addresses:
ipv4.routes:
ipv4.ignore-auto-routes:                no
ipv4.ignore-auto-dns:                   no
ipv4.dhcp-client-id:                    --
ipv4.dhcp-send-hostname:                yes
ipv4.dhcp-hostname:                     --
ipv4.never-default:                     no
ipv4.may-fail:                          yes
---------------------------------------------------------------------------
ipv6.method:                            auto
ipv6.dns:
ipv6.dns-search:
ipv6.addresses:
ipv6.routes:
ipv6.ignore-auto-routes:                no
ipv6.ignore-auto-dns:                   no
ipv6.never-default:                     no
ipv6.may-fail:                          yes
ipv6.ip6-privacy:                       2 (enabled, prefer temporary IP)
ipv6.dhcp-hostname:                     --
```

```
------------------------------------------------------------------------------
```

Pour consulter la liste profils associés à un périphérique, utilisez la commande suivante :

```
root@debian8:~# nmcli -f CONNECTIONS device show eth0
CONNECTIONS.AVAILABLE-CONNECTION-PATHS: /org/freedesktop/NetworkManager/Settings/{1,0}
CONNECTIONS.AVAILABLE-CONNECTIONS[1]:   ec7e2b5f-0d19-4a40-8639-24af88a59d79 | ip_fixe
CONNECTIONS.AVAILABLE-CONNECTIONS[2]:   0caa0d4f-0222-415a-8a4f-753ca284911e | Wired connection 1
```

Sous Debian 8, les fichiers de configuration pour le periphérique **eth0** se trouvent dans le répertoire **/etc/NetworkManager/system-connections** :

```
root@debian8:~# ls -l /etc/NetworkManager/system-connections
total 8
-rw------- 1 root root 207 Nov  3 12:00 ip_fixe
-rw------- 1 root root 170 Oct 23 16:44 Wired connection 1
```

L'étude du fichier **/etc/NetworkManager/system-connections/ip_fixe** démontre l'abscence de directives concernant les DNS :

```
root@debian8:~# cat /etc/NetworkManager/system-connections/ip_fixe
[connection]
id=ip_fixe
uuid=ec7e2b5f-0d19-4a40-8639-24af88a59d79
interface-name=eth0
type=ethernet
timestamp=1446548328

[ipv6]
method=auto

[ipv4]
method=manual
address1=10.0.2.16/24,10.0.2.2
```

La résolution des noms est donc inactive :

```
root@debian8:~# ping www.free.fr
ping: unknown host www.free.fr
```

Modifiez donc la configuration du profil **ip_fixe** :

```
root@debian8:~# nmcli connection mod ip_fixe ipv4.dns 8.8.8.8
```

Sous Debian 8 l'étude du fichier **/etc/NetworkManager/system-connections/ip_fixe** démontre que la directive concernant le serveur DNS a été ajoutée :

```
root@debian8:~# cat /etc/NetworkManager/system-connections/ip_fixe
[connection]
id=ip_fixe
uuid=ec7e2b5f-0d19-4a40-8639-24af88a59d79
interface-name=eth0
type=ethernet
timestamp=1446548328

[ipv6]
method=auto

[ipv4]
method=manual
dns=8.8.8.8;
address1=10.0.2.16/24,10.0.2.2
```

Afin que la modification du serveur DNS soit prise en compte, re-démarrez le service NetworkManager :

```
root@debian8:~# systemctl restart NetworkManager.service

root@debian8:~# systemctl status NetworkManager.service
● NetworkManager.service - Network Manager
   Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled)
```

```
   Active: active (running) since Tue 2015-11-03 12:08:26 CET; 34s ago
 Main PID: 2991 (NetworkManager)
   CGroup: /system.slice/NetworkManager.service
           └─2991 /usr/sbin/NetworkManager --no-daemon

Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> Activation (eth0) Stage 5 of 5 (IPv4 Commit) complete.
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> Activation (eth0) Stage 5 of 5 (IPv6 Commit) started...
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> Activation (eth0) Stage 5 of 5 (IPv6 Commit) complete.
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> (eth0): device state change: ip-check -> secondaries (reason
'none') [80 90 0]
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> (eth0): device state change: secondaries -> activated
(reason 'none') [90 100 0]
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> NetworkManager state is now CONNECTED_LOCAL
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> NetworkManager state is now CONNECTED_GLOBAL
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> Policy set 'eth0' (eth0) as default for IPv4 routing and
DNS.
Nov 03 12:08:26 debian8 NetworkManager[2991]: <info> Activation (eth0) successful, device activated.
Nov 03 12:08:31 debian8 NetworkManager[2991]: <info> startup complete
```

Vérifiez que le fichier **/etc/resolv.conf** ait été modifié par NetworkManager :

```
root@debian8:~# cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 8.8.8.8
```

Dernièrement vérifiez la resolution des noms :

```
root@debian8:~# ping www.free.fr
PING www.free.fr (212.27.48.10) 56(84) bytes of data.
64 bytes from www.free.fr (212.27.48.10): icmp_seq=1 ttl=63 time=44.9 ms
^C
--- www.free.fr ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1001ms
```

```
rtt min/avg/max/mdev = 44.994/44.994/44.994/0.000 ms
```

Important : Notez qu'il existe un front-end graphique en mode texte, **nmtui**, pour configurer NetworkManager.

Ajouter une Deuxième Adresse IP à un Profil

Pour ajouter une deuxième adresse IP à un profil sous Debian 8, il convient d'utiliser la commande suivante :

```
root@debian8:~# nmcli connection mod ip_fixe +ipv4.addresses 192.168.1.2/24
```

Redémarrez la machine virtuelle puis en tant que root saisissez la commande suivante :

```
root@debian8:~# nmcli connection show ip_fixe
connection.id:                          ip_fixe
connection.uuid:                        ec7e2b5f-0d19-4a40-8639-24af88a59d79
connection.interface-name:              eth0
connection.type:                        802-3-ethernet
connection.autoconnect:                 yes
connection.timestamp:                   1446560745
connection.read-only:                   no
connection.permissions:
connection.zone:                        --
connection.master:                      --
connection.slave-type:                  --
connection.secondaries:
connection.gateway-ping-timeout:        0
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          yes
```

```
802-3-ethernet.mac-address:             --
802-3-ethernet.cloned-mac-address:      --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:                     auto
802-3-ethernet.s390-subchannels:
802-3-ethernet.s390-nettype:            --
802-3-ethernet.s390-options:
ipv4.method:                            manual
ipv4.dns:                               8.8.8.8
ipv4.dns-search:
ipv4.addresses:                         { ip = 10.0.2.16/24, gw = 10.0.2.2 }; { ip = 192.168.1.2/24, gw = 0.0.0.0
}
ipv4.routes:
ipv4.ignore-auto-routes:                no
ipv4.ignore-auto-dns:                   no
ipv4.dhcp-client-id:                    --
ipv4.dhcp-send-hostname:                yes
ipv4.dhcp-hostname:                     --
ipv4.never-default:                     no
ipv4.may-fail:                          yes
ipv6.method:                            auto
ipv6.dns:
ipv6.dns-search:
ipv6.addresses:
ipv6.routes:
ipv6.ignore-auto-routes:                no
ipv6.ignore-auto-dns:                   no
ipv6.never-default:                     no
ipv6.may-fail:                          yes
ipv6.ip6-privacy:                       -1 (unknown)
ipv6.dhcp-hostname:                     --
GENERAL.NAME:                           ip_fixe
GENERAL.UUID:                           ec7e2b5f-0d19-4a40-8639-24af88a59d79
GENERAL.DEVICES:                        eth0
```

```
GENERAL.STATE:                          activated
GENERAL.DEFAULT:                        yes
GENERAL.DEFAULT6:                       no
GENERAL.VPN:                            no
GENERAL.ZONE:                           --
GENERAL.DBUS-PATH:                      /org/freedesktop/NetworkManager/ActiveConnection/1
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/Settings/1
GENERAL.SPEC-OBJECT:                    --
GENERAL.MASTER-PATH:                    --
IP4.ADDRESS[1]:                         ip = 10.0.2.16/24, gw = 10.0.2.2
IP4.ADDRESS[2]:                         ip = 192.168.1.2/24, gw = 10.0.2.2
IP4.ROUTE[1]:                           dst = 169.254.0.0/16, nh = 0.0.0.0, mt = 1000
IP4.DNS[1]:                             8.8.8.8
IP6.ADDRESS[1]:                         ip = fe80::a00:27ff:fef2:8c8f/64, gw = ::
```

Important : Notez l'ajout de la ligne **IP4.ADDRESS[2]:**.

Sous Debian 8, consultez maintenant le contenu du fichier **/etc/NetworkManager/system-connections/ip_fixe** :

```
[root@debian8:~# cat /etc/NetworkManager/system-connections/ip_fixe
[connection]
id=ip_fixe
uuid=ec7e2b5f-0d19-4a40-8639-24af88a59d79
interface-name=eth0
type=ethernet
timestamp=1446551688

[ipv6]
method=auto

[ipv4]
```

```
method=manual
dns=8.8.8.8;
address1=10.0.2.16/24,10.0.2.2
address2=192.168.1.2/24
```

Important : Notez l'ajout de la ligne **address2=192.168.1.2/24**.

La Commande hostname

La procédure de la modification du hostname est simplifiée et sa prise en compte est immédiate :

```
root@debian8:~# nmcli general hostname debian.fenestros.loc

root@debian8:~# hostname
debian.fenestros.loc

root@debian8:~# cat /etc/hostname
debian.fenestros.loc

root@debian8:~# nmcli general hostname debian8.fenestros.loc

root@debian8:~# hostname
debian8.fenestros.loc

root@debian8:~# cat /etc/hostname
debian8.fenestros.loc
```

La Commande ip

Sous Debian 8 la commande **ip** est préférée par rapport à la commande ifconfig :

```
root@debian8:~# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:f2:8c:8f brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.16/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 192.168.1.2/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef2:8c8f/64 scope link
       valid_lft forever preferred_lft forever
```

Options de la Commande ip

Les options de cette commande sont :

```
root@debian8:~# ip --help
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | addr | addrlabel | route | rule | neigh | ntable |
                   tunnel | tuntap | maddr | mroute | mrule | monitor | xfrm |
                   netns | l2tp | tcp_metrics | token | netconf }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -f[amily] { inet | inet6 | ipx | dnet | bridge | link } |
```

```
                    -4 | -6 | -I | -D | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } |
                    -o[neline] | -t[imestamp] | -b[atch] [filename] |
                    -rc[vbuf] [size]}
```

Activer/Désactiver une Interface Manuellement

Deux commandes existent pour désactiver et activer manuellement une interface réseau :

```
root@debian8:~# nmcli device disconnect eth0
root@debian8:~# nmcli device connect eth0
```

Routage Statique

La commande ip

Sous Debian 8, pour supprimer la route vers le réseau 192.168.1.0 il convient d'utiliser la commande ip et non pas la commande route :

```
root@debian8:~# ip route
default via 10.0.2.2 dev enp0s3  proto static  metric 100
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.16  metric 100
192.168.1.0/24 dev enp0s3  proto kernel  scope link  src 192.168.1.2  metric 100

root@debian8:~# ip route del 192.168.1.0/24 via 0.0.0.0

root@debian8:~# ip route
default via 10.0.2.2 dev enp0s3  proto static  metric 100
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.16  metric 100
```

Pour ajouter la route vers le réseau 192.168.1.0 :

```
root@debian8:~# ip route add 192.168.1.0/24 via 10.0.2.2

root@debian8:~# ip route
default via 10.0.2.2 dev enp0s3  proto static  metric 100
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.16  metric 100
192.168.1.0/24 via 10.0.2.2 dev enp0s3
```

La commande utilisée pour ajouter une passerelle par défaut prend la forme suivante **ip route add default via *adresse ip***.

Activer/désactiver le routage sur le serveur

Pour activer le routage sur le serveur, il convient d'activer la retransmission des paquets:

```
root@debian8:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@debian8:~# cat /proc/sys/net/ipv4/ip_forward
1
```

Pour désactiver le routage sur le serveur, il convient de désactiver la retransmission des paquets:

```
root@debian8:~# echo 0 > /proc/sys/net/ipv4/ip_forward
root@debian8:~# cat /proc/sys/net/ipv4/ip_forward
0
```

Diagnostique du Réseau

ping

Pour tester l'accessibilité d'une machine, vous devez utiliser la commande **ping** :

```
root@debian8:~# ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=0.153 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=0.137 ms
^C
--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.095/0.128/0.153/0.026 ms
```

Options de la commande ping

Les options de cette commande sont :

```
root@debian8:~# ping --help
ping: invalid option -- '-'
Usage: ping [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination
```

netstat -i

Pour visualiser les statistiques réseaux, vous disposez de la commande **netstat** :

```
root@debian8:~# netstat -i
```

```
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0       489      0      0 0           596      0      0      0 BMRU
lo        65536 0     28623      0      0 0         28623      0      0      0 LRU
```

Options de la commande netstat

Les options de cette commande sont :

```
root@debian8:~# netstat --help
usage: netstat [-vWeenNcCF] [<Af>] -r         netstat {-V|--version|-h|--help}
       netstat [-vWnNcaeol] [<Socket> ...]
       netstat { [-vWeenNac] -i | [-cWnNe] -M | -s }

        -r, --route              display routing table
        -i, --interfaces         display interface table
        -g, --groups             display multicast group memberships
        -s, --statistics         display networking statistics (like SNMP)
        -M, --masquerade         display masqueraded connections

        -v, --verbose            be verbose
        -W, --wide               don't truncate IP addresses
        -n, --numeric            don't resolve names
        --numeric-hosts          don't resolve host names
        --numeric-ports          don't resolve port names
        --numeric-users          don't resolve user names
        -N, --symbolic           resolve hardware names
        -e, --extend             display other/more information
        -p, --programs           display PID/Program name for sockets
        -c, --continuous         continuous listing

        -l, --listening          display listening server sockets
        -a, --all, --listening   display all sockets (default: connected)
```

```
       -o, --timers             display timers
       -F, --fib                display Forwarding Information Base (default)
       -C, --cache              display routing cache instead of FIB

  <Socket>={-t|--tcp} {-u|--udp} {-w|--raw} {-x|--unix} --ax25 --ipx --netrom
  <AF>=Use '-6|-4' or '-A <af>' or '--<af>'; default: inet
  List of possible address families (which support routing):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
    x25 (CCITT X.25)
```

La commande traceroute

La commande ping est à la base de la commande **traceroute**. Cette commande sert à découvrir la route empruntée pour accéder à un site donné :

```
root@debian8:~# traceroute www.i2tch.eu
traceroute to www.i2tch.eu (217.160.122.33), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.091 ms  0.056 ms  0.070 ms
 2  192.168.0.1 (192.168.0.1)  0.260 ms  0.190 ms  0.205 ms
 3  * * *
 4  195-132-10-137.rev.numericable.fr (195.132.10.137)  15.769 ms  15.711 ms  15.646 ms
 5  ip-190.net-80-236-8.asnieres.rev.numericable.fr (80.236.8.190)  15.581 ms  27.137 ms  27.053 ms
 6  ip-185.net-80-236-8.asnieres.rev.numericable.fr (80.236.8.185)  27.070 ms  10.356 ms  10.184 ms
 7  172.19.132.146 (172.19.132.146)  16.374 ms  16.357 ms  16.265 ms
 8  oneandone.franceix.net (37.49.236.42)  25.843 ms  25.539 ms  25.280 ms
 9  ae-8-0.bb-a.bap.rhr.de.oneandone.net (212.227.120.42)  21.347 ms  21.291 ms  25.058 ms
10  * * *
11  * * *
12  * * *
13  * * *
14  *^C
```

Options de la commande traceroute

Les options de cette commande sont :

```
root@debian8:~# traceroute --help
Usage:
  traceroute [ -46dFITnreAUDV ] [ -f first_ttl ] [ -g gate,... ] [ -i device ] [ -m max_ttl ] [ -N squeries ] [ -
p port ] [ -t tos ] [ -l flow_label ] [ -w waittime ] [ -q nqueries ] [ -s src_addr ] [ -z sendwait ] [ --
fwmark=num ] host [ packetlen ]
Options:
  -4                          Use IPv4
  -6                          Use IPv6
  -d  --debug                 Enable socket level debugging
  -F  --dont-fragment         Do not fragment packets
  -f first_ttl  --first=first_ttl
                              Start from the first_ttl hop (instead from 1)
  -g gate,...  --gateway=gate,...
                              Route packets through the specified gateway
                              (maximum 8 for IPv4 and 127 for IPv6)
  -I  --icmp                  Use ICMP ECHO for tracerouting
  -T  --tcp                   Use TCP SYN for tracerouting (default port is 80)
  -i device  --interface=device
                              Specify a network interface to operate with
  -m max_ttl  --max-hops=max_ttl
                              Set the max number of hops (max TTL to be
                              reached). Default is 30
  -N squeries  --sim-queries=squeries
                              Set the number of probes to be tried
                              simultaneously (default is 16)
  -n                          Do not resolve IP addresses to their domain names
  -p port  --port=port        Set the destination port to use. It is either
                              initial udp port value for "default" method
                              (incremented by each probe, default is 33434), or
```

```
                              initial seq for "icmp" (incremented as well,
                              default from 1), or some constant destination
                              port for other methods (with default of 80 for
                              "tcp", 53 for "udp", etc.)
-t tos  --tos=tos             Set the TOS (IPv4 type of service) or TC (IPv6
                              traffic class) value for outgoing packets
-l flow_label  --flowlabel=flow_label
                              Use specified flow_label for IPv6 packets
-w waittime  --wait=waittime
                              Set the number of seconds to wait for response to
                              a probe (default is 5.0). Non-integer (float
                              point) values allowed too
-q nqueries  --queries=nqueries
                              Set the number of probes per each hop. Default is
                              3
-r                            Bypass the normal routing and send directly to a
                              host on an attached network
-s src_addr  --source=src_addr
                              Use source src_addr for outgoing packets
-z sendwait  --sendwait=sendwait
                              Minimal time interval between probes (default 0).
                              If the value is more than 10, then it specifies a
                              number in milliseconds, else it is a number of
                              seconds (float point values allowed too)
-e  --extensions              Show ICMP extensions (if present), including MPLS
-A  --as-path-lookups         Perform AS path lookups in routing registries and
                              print results directly after the corresponding
                              addresses
-M name  --module=name        Use specified module (either builtin or external)
                              for traceroute operations. Most methods have
                              their shortcuts (`-I' means `-M icmp' etc.)
-O OPTS,...  --options=OPTS,...
                              Use module-specific option OPTS for the
                              traceroute module. Several OPTS allowed,
```

```
                              separated by comma. If OPTS is "help", print info
                              about available options
  --sport=num                 Use source port num for outgoing packets. Implies
                              `-N 1'
  --fwmark=num                Set firewall mark for outgoing packets
  -U  --udp                   Use UDP to particular port for tracerouting
                              (instead of increasing the port per each probe),
                              default port is 53
  -UL                         Use UDPLITE for tracerouting (default dest port
                              is 53)
  -D  --dccp                  Use DCCP Request for tracerouting (default port
                              is 33434)
  -P prot  --protocol=prot    Use raw packet of protocol prot for tracerouting
  --mtu                       Discover MTU along the path being traced. Implies
                              `-F -N 1'
  --back                      Guess the number of hops in the backward path and
                              print if it differs
  -V  --version               Print version info and exit
  --help                      Read this help and exit

Arguments:
+     host          The host to traceroute to
      packetlen     The full packet length (default is the length of an IP
                    header plus 40). Can be ignored or increased to a minimal
                    allowed value
```

Connexions à Distance

Telnet

La commande **telnet** est utilisée pour établir une connexion à distance avec un serveur telnet :

```
  # telnet numero_ip
```

> Le service telnet revient à une redirection des canaux standards d'entrée et de sortie. Notez que la connexion n'est **pas** sécurisée. Pour fermer la connexion, il faut saisir la commande **exit**. La commande telnet n'offre pas de services de transfert de fichiers. Pour cela, il convient d'utiliser la command **ftp**.

Options de la commande telnet

Les options de cette commande sont :

```
root@debian8:~# telnet --help
telnet: invalid option -- '-'
Usage: telnet [-4] [-6] [-8] [-E] [-L] [-a] [-d] [-e char] [-l user]
    [-n tracefile] [ -b addr ] [-r] [host-name [port]]
```

wget

La commande **wget** est utilisée pour récupérer un fichier via http ou ftp :

```
root@debian8:~# wget https://www.dropbox.com/s/li5tyou8msofuwb/fichier_test?dl=0
--2017-06-22 15:58:28--  https://www.dropbox.com/s/li5tyou8msofuwb/fichier_test?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location:
https://dl.dropboxusercontent.com/content_link/DWSZfzGE6aRiRjGhALuA1VZTLwR9OLGZmTFKef0Cq1PdVJUuIXxfApaV4D13DopI/f
ile [following]
--2017-06-22 15:58:30--
```

```
https://dl.dropboxusercontent.com/content_link/DWSZfzGE6aRiRjGhALuA1VZTLwR9OLGZmTFKef0Cq1PdVJUuIXxfApaV4D13DopI/f
ile
Resolving dl.dropboxusercontent.com (dl.dropboxusercontent.com)... 162.125.65.6
Connecting to dl.dropboxusercontent.com (dl.dropboxusercontent.com)|162.125.65.6|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17 [text/plain]
Saving to: ‘fichier_test?dl=0’

fichier_test?dl=0   100%[====================>]      17  --.-KB/s   in 0s

2017-06-22 15:58:31 (3.53 MB/s) - ‘fichier_test?dl=0’ saved [17/17]
```

Options de la commande wget

Les options de cette commande sont :

```
root@debian8:~# wget --help
GNU Wget 1.16, a non-interactive network retriever.
Usage: wget [OPTION]... [URL]...

Mandatory arguments to long options are mandatory for short options too.

Startup:
  -V,  --version                   display the version of Wget and exit.
  -h,  --help                      print this help.
  -b,  --background                go to background after startup.
  -e,  --execute=COMMAND           execute a `.wgetrc'-style command.

Logging and input file:
  -o,  --output-file=FILE          log messages to FILE.
  -a,  --append-output=FILE        append messages to FILE.
  -d,  --debug                     print lots of debugging information.
  -q,  --quiet                     quiet (no output).
```

```
  -v,  --verbose                   be verbose (this is the default).
  -nv, --no-verbose                turn off verboseness, without being quiet.
       --report-speed=TYPE         Output bandwidth as TYPE.  TYPE can be bits.
  -i,  --input-file=FILE           download URLs found in local or external FILE.
  -F,  --force-html                treat input file as HTML.
  -B,  --base=URL                  resolves HTML input-file links (-i -F)
                                   relative to URL.
       --config=FILE               Specify config file to use.
       --no-config                 Do not read any config file.

Download:
  -t,  --tries=NUMBER              set number of retries to NUMBER (0 unlimits).
       --retry-connrefused         retry even if connection is refused.
  -O,  --output-document=FILE      write documents to FILE.
  -nc, --no-clobber                skip downloads that would download to
                                   existing files (overwriting them).
  -c,  --continue                  resume getting a partially-downloaded file.
       --start-pos=OFFSET          start downloading from zero-based position OFFSET.
       --progress=TYPE             select progress gauge type.
       --show-progress             display the progress bar in any verbosity mode.
  -N,  --timestamping              don't re-retrieve files unless newer than
                                   local.
  --no-use-server-timestamps       don't set the local file's timestamp by
                                   the one on the server.
  -S,  --server-response           print server response.
       --spider                    don't download anything.
  -T,  --timeout=SECONDS           set all timeout values to SECONDS.
       --dns-timeout=SECS          set the DNS lookup timeout to SECS.
       --connect-timeout=SECS      set the connect timeout to SECS.
       --read-timeout=SECS         set the read timeout to SECS.
  -w,  --wait=SECONDS              wait SECONDS between retrievals.
       --waitretry=SECONDS         wait 1..SECONDS between retries of a retrieval.
       --random-wait               wait from 0.5*WAIT...1.5*WAIT secs between retrievals.
       --no-proxy                  explicitly turn off proxy.
```

```
  -Q,  --quota=NUMBER              set retrieval quota to NUMBER.
       --bind-address=ADDRESS      bind to ADDRESS (hostname or IP) on local host.
       --limit-rate=RATE           limit download rate to RATE.
       --no-dns-cache              disable caching DNS lookups.
       --restrict-file-names=OS    restrict chars in file names to ones OS allows.
       --ignore-case               ignore case when matching files/directories.
  -4,  --inet4-only                connect only to IPv4 addresses.
  -6,  --inet6-only                connect only to IPv6 addresses.
       --prefer-family=FAMILY      connect first to addresses of specified family,
                                   one of IPv6, IPv4, or none.
       --user=USER                 set both ftp and http user to USER.
       --password=PASS             set both ftp and http password to PASS.
       --ask-password              prompt for passwords.
       --no-iri                    turn off IRI support.
       --local-encoding=ENC        use ENC as the local encoding for IRIs.
       --remote-encoding=ENC       use ENC as the default remote encoding.
       --unlink                    remove file before clobber.

Directories:
  -nd, --no-directories            don't create directories.
  -x,  --force-directories         force creation of directories.
  -nH, --no-host-directories       don't create host directories.
       --protocol-directories      use protocol name in directories.
  -P,  --directory-prefix=PREFIX   save files to PREFIX/...
       --cut-dirs=NUMBER           ignore NUMBER remote directory components.

HTTP options:
       --http-user=USER            set http user to USER.
       --http-password=PASS        set http password to PASS.
       --no-cache                  disallow server-cached data.
       --default-page=NAME         Change the default page name (normally
                                   this is `index.html'.).
  -E,  --adjust-extension          save HTML/CSS documents with proper extensions.
       --ignore-length             ignore `Content-Length' header field.
```

```
       --header=STRING             insert STRING among the headers.
       --max-redirect              maximum redirections allowed per page.
       --proxy-user=USER           set USER as proxy username.
       --proxy-password=PASS       set PASS as proxy password.
       --referer=URL               include `Referer: URL' header in HTTP request.
       --save-headers              save the HTTP headers to file.
  -U,  --user-agent=AGENT          identify as AGENT instead of Wget/VERSION.
       --no-http-keep-alive        disable HTTP keep-alive (persistent connections).
       --no-cookies                don't use cookies.
       --load-cookies=FILE         load cookies from FILE before session.
       --save-cookies=FILE         save cookies to FILE after session.
       --keep-session-cookies      load and save session (non-permanent) cookies.
       --post-data=STRING          use the POST method; send STRING as the data.
       --post-file=FILE            use the POST method; send contents of FILE.
       --method=HTTPMethod         use method "HTTPMethod" in the request.
       --body-data=STRING          Send STRING as data. --method MUST be set.
       --body-file=FILE            Send contents of FILE. --method MUST be set.
       --content-disposition       honor the Content-Disposition header when
                                   choosing local file names (EXPERIMENTAL).
       --content-on-error          output the received content on server errors.
       --auth-no-challenge         send Basic HTTP authentication information
                                   without first waiting for the server's
                                   challenge.

HTTPS (SSL/TLS) options:
       --secure-protocol=PR        choose secure protocol, one of auto, SSLv2,
                                   SSLv3, TLSv1 and PFS.
       --https-only                only follow secure HTTPS links
       --no-check-certificate      don't validate the server's certificate.
       --certificate=FILE          client certificate file.
       --certificate-type=TYPE     client certificate type, PEM or DER.
       --private-key=FILE          private key file.
       --private-key-type=TYPE     private key type, PEM or DER.
       --ca-certificate=FILE       file with the bundle of CA's.
```

```
       --ca-directory=DIR          directory where hash list of CA's is stored.
       --random-file=FILE          file with random data for seeding the SSL PRNG.
       --egd-file=FILE             file naming the EGD socket with random data.

FTP options:
       --ftp-user=USER             set ftp user to USER.
       --ftp-password=PASS         set ftp password to PASS.
       --no-remove-listing         don't remove `.listing' files.
       --no-glob                   turn off FTP file name globbing.
       --no-passive-ftp            disable the "passive" transfer mode.
       --preserve-permissions      preserve remote file permissions.
       --retr-symlinks             when recursing, get linked-to files (not dir).

WARC options:
       --warc-file=FILENAME        save request/response data to a .warc.gz file.
       --warc-header=STRING        insert STRING into the warcinfo record.
       --warc-max-size=NUMBER      set maximum size of WARC files to NUMBER.
       --warc-cdx                  write CDX index files.
       --warc-dedup=FILENAME       do not store records listed in this CDX file.
       --no-warc-compression       do not compress WARC files with GZIP.
       --no-warc-digests           do not calculate SHA1 digests.
       --no-warc-keep-log          do not store the log file in a WARC record.
       --warc-tempdir=DIRECTORY    location for temporary files created by the
                                   WARC writer.

Recursive download:
  -r,  --recursive                 specify recursive download.
  -l,  --level=NUMBER              maximum recursion depth (inf or 0 for infinite).
       --delete-after              delete files locally after downloading them.
  -k,  --convert-links             make links in downloaded HTML or CSS point to
                                   local files.
       --backups=N                 before writing file X, rotate up to N backup files.
  -K,  --backup-converted          before converting file X, back up as X.orig.
  -m,  --mirror                    shortcut for -N -r -l inf --no-remove-listing.
```

```
  -p,  --page-requisites           get all images, etc. needed to display HTML page.
       --strict-comments           turn on strict (SGML) handling of HTML comments.

Recursive accept/reject:
  -A,  --accept=LIST               comma-separated list of accepted extensions.
  -R,  --reject=LIST               comma-separated list of rejected extensions.
       --accept-regex=REGEX        regex matching accepted URLs.
       --reject-regex=REGEX        regex matching rejected URLs.
       --regex-type=TYPE           regex type (posix).
  -D,  --domains=LIST              comma-separated list of accepted domains.
       --exclude-domains=LIST      comma-separated list of rejected domains.
       --follow-ftp                follow FTP links from HTML documents.
       --follow-tags=LIST          comma-separated list of followed HTML tags.
       --ignore-tags=LIST          comma-separated list of ignored HTML tags.
  -H,  --span-hosts                go to foreign hosts when recursive.
  -L,  --relative                  follow relative links only.
  -I,  --include-directories=LIST  list of allowed directories.
       --trust-server-names        use the name specified by the redirection
                                   url last component.
  -X,  --exclude-directories=LIST  list of excluded directories.
  -np, --no-parent                 don't ascend to the parent directory.

Mail bug reports and suggestions to <bug-wget@gnu.org>.
```

ftp

La commande **ftp** est utilisée pour le transfert de fichiers:

```
root@debian8:~# ftp ftp2.fenestros.com
Connected to anonymous.ftp.ovh.net.
220 anonymous.ftp.ovh.net NcFTPd Server (licensed copy) ready.
Name (ftp2.fenestros.com:trainee): anonymous
331 Guest login ok, send your complete e-mail address as password.
```

```
Password:
230 Logged in anonymously.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Une fois connecté, il convient d'utiliser la commande **help** pour afficher la liste des commandes disponibles :

```
ftp> help
Commands may be abbreviated.  Commands are:

!       dir         mdelete         qc         site
$       disconnect    mdir          sendport    size
account     exit        mget          put         status
append      form        mkdir         pwd         struct
ascii       get         mls           quit        system
bell        glob        mode          quote         sunique
binary      hash        modtime         recv        tenex
bye     help         mput         reget         tick
case        idle        newer         rstatus        trace
cd      image         nmap         rhelp         type
cdup        ipany         nlist         rename         user
chmod       ipv4        ntrans         reset        umask
close       ipv6        open        restart        verbose
cr      lcd         prompt         rmdir         ?
delete      ls          passive         runique
debug       macdef         proxy         send
```

Le caractère **!** permet d'exécuter une commande sur la machine cliente

```
ftp> !pwd
/root
```

Pour transférer un fichier vers le serveur, il convient d'utiliser la commande **put** :

```
ftp> put nom_fichier_local nom_fichier_distant
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mput**. Dans ce cas précis, il convient de saisir la commande suivante:

```
ftp> mput nom*.*
```

Pour transférer un fichier du serveur, il convient d'utiliser la commande **get** :

```
ftp> get nom_fichier
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mget** (voir la commande **mput** ci-dessus).

Pour supprimer un fichier sur le serveur, il convient d'utiliser la commande **del** :

```
ftp> del nom_fichier
```

Pour fermer la session, il convient d'utiliser la commande **quit** :

```
ftp> quit
221 Goodbye.
root@debian8:~#
```

SSH

Introduction

La commande **ssh** est le successeur et la remplaçante de la commande **rlogin**. Il permet d'établir des connexions sécurisées avec une machine distante. SSH comporte cinq acteurs :

- Le **serveur SSH**
  - le démon sshd, qui s'occupe des authentifications et autorisations des clients,

- Le **client SSH**
  - ssh ou scp, qui assure la connexion et le dialogue avec le serveur,
- La **session** qui représente la connexion courante et qui commence juste après l'authentification réussie,
- Les **clefs**
  - **Couple de clef utilisateur asymétriques** et persistantes qui assurent l'identité d'un utilisateur et qui sont stockés sur disque dur,
  - **Clef hôte asymétrique et persistante** garantissant l'identité du serveur er qui est conservé sur disque dur
  - **Clef serveur asymétrique et temporaire** utilisée par le protocole SSH1 qui sert au chiffrement de la clé de session,
  - **Clef de session symétrique qui est générée aléatoirement** et qui permet le chiiffrement de la communication entre le client et le serveur. Elle est détruite en fin de session. SSH-1 utilise une seule clef tandis que SSH-2 utilise une clef par direction de la communication,
- La **base de données des hôtes connus** qui stocke les clés des connexions précédentes.

SSH fonctionne de la manière suivante pour la la mise en place d'un canal sécurisé:

- Le client contacte le serveur sur son port 22,
- Les client et le serveur échangent leur version de SSH. En cas de non-compatibilité de versions, l'un des deux met fin au processus,
- Le serveur SSH s'identifie auprès du client en lui fournissant :
  - Sa clé hôte,
  - Sa clé serveur,
  - Une séquence aléatoire de huit octets à inclure dans les futures réponses du client,
  - Une liste de méthodes de chiffrage, compression et authentification,
- Le client et le serveur produisent un identifiant identique, un haché MD5 long de 128 bits contenant la clé hôte, la clé serveur et la séquence aléatoire,
- Le client génère sa clé de session symétrique et la chiffre deux fois de suite, une fois avec la clé hôte du serveur et la deuxième fois avec la clé serveur. Le client envoie cette clé au serveur accompagnée de la séquence aléatoire et un choix d'algorithmes supportés,
- Le serveur déchiffre la clé de session,
- Le client et le serveur mettent en place le canal sécurisé.

SSH-1

SSH-1 utilise une paire de clefs de type RSA1. Il assure l'intégrité des données par une **Contrôle de Redondance Cyclique** (CRC) et est un bloc dit **monolithique**.

Afin de s'identifier, le client essaie chacune des six méthodes suivantes :

- **Kerberos**,
- **Rhosts**,
- **RhostsRSA**,
- Par **clef asymétrique**,
- **TIS**,
- Par **mot de passe**.

SSH-2

SSH-2 utilise **DSA** ou **RSA**. Il assure l'intégrité des données par l'algorithme **HMAC**. SSH-2 est organisé en trois **couches** :

- **SSH-TRANS** – Transport Layer Protocol,
- **SSH-AUTH** – Authentification Protocol,
- **SSH-CONN** – Connection Protocol.

SSH-2 diffère de SSH-1 essentiellement dans la phase authentification.

Trois méthodes d'authentification :

- Par **clef asymétrique**,
  - Identique à SSH-1 sauf avec l'algorithme DSA,
- **RhostsRSA**,
- Par **mot de passe**.

Options de la commande

Les options de cette commande sont :

```
root@debian8:~# ssh --help
unknown option -- -
usage: ssh [-1246AaCfgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
           [-D [bind_address:]port] [-E log_file] [-e escape_char]
```

```
           [-F configfile] [-I pkcs11] [-i identity_file]
           [-L [bind_address:]port:host:hostport] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-p port]
           [-Q cipher | cipher-auth | mac | kex | key]
           [-R [bind_address:]port:host:hostport] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] [user@]hostname [command]
```

L'authentification par mot de passe

L'utilisateur fournit un mot de passe au client ssh. Le client ssh le transmet de façon sécurisée au serveur ssh puis le serveur vérifie le mot de passe et l'accepte ou non.

Avantage:

- Aucune configuration de clef asymétrique n'est nécessaire.

Inconvénients:

- L'utilisateur doit fournir à chaque connexion un identifiant et un mot de passe,
- Moins sécurisé qu'un système par clef asymétrique.

L'authentification par clef asymétrique

- Le **client** envoie au serveur une requête d'authentification par clé asymétrique qui contient le module de la clé à utiliser,
- Le **serveur** recherche une correspondance pour ce module dans le fichier des clés autorisés **~/.ssh/authorized_keys**,
  - Dans le cas où une correspondance n'est pas trouvée, le serveur met fin à la communication,
  - Dans le cas contraire le serveur génère une chaîne aléatoire de 256 bits appelée un **challenge** et la chiffre avec la **clé publique du client**,
- Le **client** reçoit le challenge et le décrypte avec la partie privée de sa clé. Il combine le challenge avec l'identifiant de session et chiffre le résultat. Ensuite il envoie le résultat chiffré au serveur.
- Le **serveur** génère le même haché et le compare avec celui reçu du client. Si les deux hachés sont identiques, l'authentification est réussie.

Installation

Pour installer/mettre à jour le serveur **sshd**, utilisez **apt-get** :

```
root@debian8:~# which sshd
/usr/sbin/sshd
root@debian8:~# which ssh
/usr/bin/ssh
root@debian8:~# dpkg -S /usr/sbin/sshd
openssh-server: /usr/sbin/sshd
root@debian8:~# dpkg -S /usr/bin/ssh
openssh-client: /usr/bin/ssh
root@debian8:~# apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Important - Pour les stations de travail, installez le client : **openssh-client**.

Options de la commande

Les options de la commande sont :

```
SYNOPSIS
     sshd [-46DdeiqTt] [-b bits] [-C connection_spec] [-f config_file] [-g login_grace_time] [-h host_key_file]
[-k key_gen_time] [-o option] [-p port]  [-u len]
```

Configuration

> **Important** - La configuration doit s'effectuer dans la fenêtre de la VM sous VirtualBox. Les connexions en ssh doivent de faire à partir d'un terminal ou de l'application putty.

Serveur

La configuration du serveur s'effectue dans le fichier **/etc/ssh/sshd_config** :

```
root@debian8:~# cat /etc/ssh/sshd_config
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024
```

```
# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin without-password
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes

# Kerberos options
#KerberosAuthentication no
```

```
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes

X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes
```

Pour ôter les lignes de commentaires dans ce fichier, utilisez la commande suivante :

```
root@debian8:~# cd /tmp ; grep -E -v '^(#|$)'  /etc/ssh/sshd_config > sshd_config
root@debian8:/tmp# cat sshd_config
Port 22
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
UsePrivilegeSeparation yes
KeyRegenerationInterval 3600
ServerKeyBits 1024
SyslogFacility AUTH
LogLevel INFO
LoginGraceTime 120
PermitRootLogin without-password
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
```

Pour sécuriser le serveur ssh, ajoutez ou modifiez les directives suivantes :

```
AllowGroups adm
Banner /etc/issue.net
PermitRootLogin no
X11Forwarding no
```

Votre fichier ressemblera à celui-ci :

```
AllowGroups adm
Banner /etc/issue.net
PermitRootLogin no
X11Forwarding no
Port 22
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
UsePrivilegeSeparation yes
KeyRegenerationInterval 3600
ServerKeyBits 1024
SyslogFacility AUTH
LogLevel INFO
LoginGraceTime 120
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
X11DisplayOffset 10
```

```
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
```

A Faire - Renommez le fichier **/etc/ssh/sshd_config** en **/etc/ssh/sshd_config.old** puis copiez le fichier **/tmp/sshd_config** vers **/etc/sshd/**. Redémarrez ensuite le service sshd. N'oubliez pas de mettre l'utilisateur **trainee** dans le groupe **adm** !

Saisissez maintenant les commandes suivantes en tant que **trainee** :

Lors de la génération des clefs, la passphrase doit être **vide**.

```
trainee@debian8:~$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_dsa):
Created directory '/home/trainee/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_dsa.
Your public key has been saved in /home/trainee/.ssh/id_dsa.pub.
The key fingerprint is:
07:c9:8f:e3:b9:c9:26:8d:49:a9:22:08:d2:6e:22:4d trainee@debian8
The key's randomart image is:
+---[DSA 1024]----+
|                 |
|       . .       |
```

```
|        +        |
|         +       |
| .     .S o       |
|o E   o. +       |
|++   o +o        |
|= = . +.oo       |
|.+ .   o+        |
+-----------------+
trainee@debian8:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_rsa.
Your public key has been saved in /home/trainee/.ssh/id_rsa.pub.
The key fingerprint is:
75:ed:8b:ed:32:be:78:72:19:88:12:d1:4d:56:7a:d2 trainee@debian8
The key's randomart image is:
+---[RSA 2048]----+
|       . oo..    |
|      . ...o .   |
|       .  + E .  |
|      .  . + .   |
|       .S. .  .  |
|      . . . .o . |
|       .    .oo  |
|          ..*.   |
|          .=o+.  |
+-----------------+
trainee@debian8:~$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Your identification has been saved in /home/trainee/.ssh/id_ecdsa.
Your public key has been saved in /home/trainee/.ssh/id_ecdsa.pub.
The key fingerprint is:
66:70:a4:c5:99:3e:4f:97:b3:8d:3a:87:4c:de:c7:8c trainee@debian8
The key's randomart image is:
+---[ECDSA 256]---+
|       .oo       |
|       ++        |
|      o..    .   |
|       oo . +    |
|        S+ . =   |
|       o  o o .  |
|         + + +   |
|          * E +  |
|           o .   |
+-----------------+
trainee@debian8:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_ed25519.
Your public key has been saved in /home/trainee/.ssh/id_ed25519.pub.
The key fingerprint is:
92:e7:04:c6:77:9f:c3:46:f2:21:7d:c9:35:7d:c4:26 trainee@debian8
The key's randomart image is:
+--[ED25519 256]--+
|               ++|
|     .     . .Eo*|
|      + . + + +o.|
|     . + . B +   |
|      o S   B    |
|       =   . .   |
|        .        |
```

```
|                 |
|                 |
+-----------------+
```

Les clés générées seront placées dans le répertoire **~/.ssh/**.

Utilisation

La commande ssh prend la forme suivante:

```
ssh -l nom_de_compte numero_ip (nom_de_machine)
```

En saisissant cette commande sur votre propre machine, vous obtiendrez un résultat similaire à celle-ci :

```
trainee@debian8:~$ su -
Password: fenestros

root@debian8:~# ssh -l trainee localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is 79:00:60:0e:2b:71:5e:cb:1a:08:45:e8:ab:45:b8:dd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
trainee@localhost's password: trainee

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

```
Last login: Sun Aug  7 08:17:07 2016 from 10.0.2.2

trainee@debian8:~$ pwd
/home/trainee

trainee@debian8:~$ whoami
trainee

trainee@debian8:~$ exit
logout
Connection to localhost closed.
```

Tunnels SSH

Le protocole SSH peut être utilisé pour sécuriser les protocoles tels telnet, pop3 etc.. En effet, on peut créer un *tunnel* SSH dans lequel passe les communications du protocole non-sécurisé.

La commande pour créer un tunnel ssh prend la forme suivante :

```
ssh -N -f compte@hôte -Lport-local:localhost:port_distant
```

Dans votre cas, vous allez créer un tunnel dans votre propre vm entre le port 15023 et le port 23 :

```
root@debian8:~# ssh -N -f trainee@localhost -L15023:localhost:23
trainee@localhost's password: trainee
```

Installez maintenant le serveur telnet :

```
root@debian8:~# apt-get install telnetd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
```

```
  openbsd-inetd
The following NEW packages will be installed:
  openbsd-inetd telnetd
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 80.0 kB of archives.
After this operation, 289 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.fr.debian.org/debian/ jessie/main openbsd-inetd amd64 0.20140418-2 [35.4 kB]
Get:2 http://ftp.fr.debian.org/debian/ jessie/main telnetd amd64 0.17-36 [44.6 kB]
Fetched 80.0 kB in 0s (585 kB/s)
Selecting previously unselected package openbsd-inetd.
(Reading database ... 82450 files and directories currently installed.)
Preparing to unpack .../openbsd-inetd_0.20140418-2_amd64.deb ...
Unpacking openbsd-inetd (0.20140418-2) ...
Selecting previously unselected package telnetd.
Preparing to unpack .../telnetd_0.17-36_amd64.deb ...
Unpacking telnetd (0.17-36) ...
Processing triggers for systemd (215-17+deb8u4) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up openbsd-inetd (0.20140418-2) ...
Setting up telnetd (0.17-36) ...
Adding user telnetd to group utmp
Processing triggers for systemd (215-17+deb8u4) ...
```

Connectez-vous ensuite via telnet sur le port 15023, vous constaterez que votre connexion n'aboutit pas :

```
root@debian8:~# telnet localhost 15023
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Debian GNU/Linux 8
debian8 login: trainee
Password:
Last login: Sun Aug  7 09:30:13 BST 2016 from localhost on pts/1
```

```
Linux debian8 3.16.0-4-amd64 #1 SMP Debian 3.16.7-ckt25-2 (2016-04-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

trainee@debian8:~$ pwd
/home/trainee

trainee@debian8:~$ whoami
trainee

trainee@debian8:~$ exit
logout
Connection closed by foreign host.
```

Notez bien que votre communication telnet passe par le tunnel SSH.

SCP

Introduction

La commande **scp** est le successeur et la remplaçante de la commande **rcp** de la famille des commandes **remote**. Il permet de faire des transferts sécurisés à partir d'une machine distante :

```
$ scp compte@numero_ip(nom_de_machine):/chemin_distant/fichier_distant /chemin_local/fichier_local
```

ou vers une machine distante :

```
$ scp /chemin_local/fichier_local compte@numero_ip(nom_de_machine):/chemin_distant/fichier_distant
```

Utilisation

Nous allons maintenant utiliser **scp** pour chercher un fichier sur le «serveur» :

Créez le fichier **/home/trainee/scp_test** :

```
root@debian8:~# exit
logout

trainee@debian8:~$ pwd
/home/trainee

trainee@debian8:~$ touch scp_test
```

Récupérez le fichier **scp_test** en utilisant scp :

```
trainee@debian8:~$ scp trainee@127.0.0.1:/home/trainee/scp_test /tmp/scp_test
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is 79:00:60:0e:2b:71:5e:cb:1a:08:45:e8:ab:45:b8:dd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
trainee@127.0.0.1's password:
scp_test
100%    0     0.0KB/s   00:00
trainee@debian8:~$ ls /tmp/scp_test
/tmp/scp_test
```

Mise en place des clefs

Il convient maintenant de se connecter sur le «serveur» en utilisant ssh et vérifiez la présence du répertoire ~/.ssh :

En saisissant cette commande, vous obtiendrez une fenêtre similaire à celle-ci :

```
trainee@debian8:~$ ssh -l trainee 127.0.0.1
trainee@127.0.0.1's password: trainee

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug  7 09:36:15 2016 from localhost

trainee@debian8:~$ ls -la | grep .ssh
drwx------  2 trainee trainee 4096 Aug  7 09:38 .ssh
trainee@debian8:~$ exit
logout
Connection to 127.0.0.1 closed.
```

> Si le dossier distant .ssh n'existe pas dans le répertoire personnel de l'utilisateur connecté, il faut le créer avec des permissions de 700.
> Dans votre cas, puisque votre machine joue le rôle de serveur **et** du client, le dossier /home/trainee/.ssh existe **déjà**.

Ensuite, il convient de transférer le fichier local **.ssh/id_ecdsa.pub** du «client» vers le «serveur» en le renommant en **authorized_keys** :

```
trainee@debian8:~$ scp .ssh/id_ecdsa.pub trainee@127.0.0.1:/home/trainee/.ssh/authorized_keys
trainee@127.0.0.1's password:
id_ecdsa.pub
```

```
100%  177     0.2KB/s   00:00
```

Important : Le fichier **authorized_keys** doit avoir les permissions de 600.

Connectez-vous via telnet et insérer les clefs publiques restantes dans le fichier .ssh/authorized_keys :

```
trainee@debian8:~$ ssh -l trainee localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is 79:00:60:0e:2b:71:5e:cb:1a:08:45:e8:ab:45:b8:dd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug  7 09:39:27 2016 from localhost
trainee@debian8:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
trainee@debian8:~$ cat .ssh/id_dsa.pub >> .ssh/authorized_keys
trainee@debian8:~$ cat .ssh/id_ed25519.pub >> .ssh/authorized_keys
trainee@debian8:~$ cat .ssh/authorized_keys
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGdOSQvO1nuPdg5FqzsdwD2l6JuylCDjguGV89prpCYndSaNDyLzE8qV2lHNy
8Nk+026oDSyceTOJMJ56rzCVpM= trainee@debian8
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDdVGlmhx8NjbSOk4DHFcUU/pLkxmaLH2VaAyOiFzfgqCYPMaurc8S/GboecKsl39FV4QMGrOO1s+36/c6rL
f28kicwDiS1UcbjHJt4eo0YniMRUhx9rz0polS+AARt/vfvGdfLVkjFcMg51WVXOQ6YmLc9nGKnKcQpQ8xbqO3cPxTWOMN9V0GozvqVMa9QbTEgTJ
g9gANt32WdA70IZdMB4l4H0XgV6cI+b01z5xjER6MgNF/52qbRiaPJe0+n/ujrjcXQqQ1NpOKJwK6XHtyKElM9t+xESPGwQJ3q9h4xxZH3zlMxcnt
WiJwBm8U+Z8NLco+o+RhEDhUcLWanR3PX trainee@debian8
```

```
ssh-dss
AAAAB3NzaC1kc3MAAACBAP0CgPkPk5tkJeZC3bmB+7yuSjWzdeNF3WeKKxd6YRfHwYVtTWbpw5VMBEGQVVTLtozGmSaZXI2rcL2ZjLRKu2PrWMXMH
Nptu+5xD1B7obRutzNa0wzxRuwyI7Ducq8PK53nbm56M4GormuZmc/0MBWrVi+QazPChbPrHB1oYSNZAAAAFQD2qjLJr89nWmGT0FrM3CtZmdTTmw
AAAIEA4AtkO2+xQmqKyIKka2WJy6d2VR6KZEFdCjWXakXp0OsqotFgghsokEnGpeyUTA53ItZoL+ozV2Vb+4yskFS1RL2c/jHyc/C9j5tV9xgsSjb
3Iz77YK+7plamZQFiOWyZwpuPirlsPGHZ//p3z5+qsL7sYy1UR9UxzxSm+2eDc0MAAACBAK697cNU0Z3PbkqjPiypfAIEpTMVHmYutlnpLjU0H6Eu
Vz4Ku3SpD9iZwX9lrSMRDrTs4gd0WjMR7Ahc4zYdILC94tZDl6Uz9Obq5Y2seqCSA00fCxCi6t4rEDDT5Rpuo1JX6pNklQsGiUJyg1Z1hJvWPRZzz
sM+ZXszz9Ar5b96 trainee@debian8
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOmyQOP1WnEAbS/qkqQxvJWHvCin8CiGBm2bzOhfPU1Z trainee@debian8
```

Important : Notez que l'authentification a utilisé le couple de clefs asymétrique et aucun mot de passe n'a été requis.

Le Parefeu Netfilter

Configuration du Pare-feu Netfilter/iptables

Introduction

Netfilter est composé de 5 *hooks* :

- NF_IP_PRE_ROUTING
- NF_IP_LOCAL_IN
- NF_IP_LOCAL_OUT
- NF_IP_FORWARD
- NF_IP_POSTROUTING

Ces hooks sont utilisés par deux branches, la première est celle concernée par les paquets qui entrent vers des services locaux :

- NF_IP_PRE_ROUTING > NF_IP_LOCAL_IN > NF_IP_LOCAL_OUT > NF_IP_POSTROUTING

tandis que la deuxième concerne les paquets qui traversent la passerelle:

- NF_IP_PRE_ROUTING > NF_IP_FORWARD > NF_IP_POSTROUTING

Si IPTABLES a été compilé en tant que module, son utilisation nécessite le chargement de plusieurs modules supplémentaires en fonction de la situation:

- iptable_filter
- iptable_mangle
- iptable_net
- etc

Netfilter est organisé en **tables**. La commande **iptables** de netfilter permet d'insérer des **policies** dans les **chaines**:

- La table **FILTER**
  - La chaîne INPUT
    - Concerne les paquets entrants
      - Policies: ACCEPT, DROP, REJECT
  - La chaîne OUTPUT
    - Concerne les paquets sortants
      - Policies: ACCEPT, DROP, REJECT
  - La chaîne FORWARD
    - Concerne les paquets traversant le par-feu.
      - Policies: ACCEPT, DROP, REJECT

Si aucune table n'est précisée, c'est la table FILTER qui s'applique par défaut.

- La table **NAT**
  - La chaîne PREROUTING
    - Permet de faire la translation d'adresse de destination
      - Cibles: SNAT, DNAT, MASQUERADE
  - La chaîne POSTROUTING
    - Permet de faire la translation d'adresse de la source
      - Cibles: SNAT, DNAT, MASQUERADE
  - Le cas spécifique OUTPUT

        - Permet la modification de la destination des paquets générés localement

- La table **MANGLE**
    - Permet le marquage de paquets générés localement (OUTPUT) et entrants (PREROUTING)

Les **policies** sont:

- ACCEPT
    - Permet d'accepter le paquet concerné
- DROP
    - Permet de rejeter le paquet concerné sans générer un message d'erreur
- REJECT
    - Permet de rejeter le paquet concerné en générant une message d'erreur

Les **cibles** sont:

- SNAT
    - Permet de modifier l'adresse source du paquet concerné
- DNAT
    - Permet de modifier l'adresse de destination du paquet concerné
- MASQUERADE
    - Permet de remplacer l'adresse IP privée de l'expéditeur par un socket public de la passerelle.

IPTABLES peut être configuré soit par des outils tels shorewall, soit en utilisant des lignes de commandes ou un script. Dans ce dernier cas, la ligne prend la forme:

```
# IPTABLES --action CHAINE --option1 --option2
```

Les actions sont:

Action	Abréviation	Déscription
- -append	-A	Ajouter une règle à la fin de la chaîne spécifiée
- -delete	-D	Supprimer une règle en spécifiant son numéro ou la règle à supprimer
- -replace	-R	Permet de remplacer la règle spécifée par son numéro
- -insert	-I	Permet d'insérer une règle à l'endroit spécifié

Action	Abréviation	Déscription
- -list	-L	Permet d'afficher des règles
- -flush	-F	Permet de vider toutes les règles d'une chaîne

Les options sont:

Option	Abréviation	Déscription
- -protocol	-p	Permet de spécifier un protocol - tcp, udp, icmp, all
- -source	-s	Permet de spécifier une adresse source
- -destination	-d	Permet de spécifier une adresse de destination
- -in-interface	-i	Permet de spécifier une interface réseau d'entrée
- -out-interface	-o	Permet de spécifier une interface réseau de sortie
- -fragment	-f	Permet de ne spécifier que les paquets fragmentés
- -source-port	-sport	Permet de spécifier un port source ou une plage de ports source
- -destination-port	-dport	Permet de spécifier un port de destination ou une plage de ports de destination
- -tcp-flags	s/o	Permet de spécifier un flag TCP à matcher - SYN, ACK, FIN, RST, URG, PSH, ALL, NONE
- -icmp-type	s/o	Permet de spécifier un type de paquet ICMP
- -mac-source	s/o	Permet de spécifier une adresse MAC

Les options spécifiques à NET sont:

- -to-destination	s/o	Permet de spécifier l'adresse de destination d'une translation
- -to-source	s/o	Permet spécifier l'adresse source d'une translation

Les options spécifiques aux LOGS sont:

- -log-level	s/o	Permet de spécifier le niveau de logs
- -log-prefix	s/o	Permet de spécifier un préfix pour les logs

L'option spécifique au STATEFUL est:

- -state	s/o	Permet de spécifier l'état du paquet à vérifier

Ce dernier cas fait référence au STATEFUL. Le STATEFUL est la capacité du par-feu à enregistrer dans une table spécifique, l'état des différentes connexions. Cette table s'appelle une **table d'état**. Le principe du fonctionnement de STATEFUL est simple, à savoir, si le paquet entrant appartient à une communication déjà établie, celui-ci n'est pas vérifié.

Il existe 4 états:

- NEW
  - Le paquet concerne une nouvelle connexion et contient donc un flag SYN à 1
- ESTABLISHED
  - Le paquet concerne une connexion déjà établie. Le paquet ne doit contenir **ni** flag SYN à 1, **ni** flag FIN à 1
- RELATED
  - Le paquet est d'une connexion qui présente une relation avec une autre connexion
- INVALID
  - La paquet provient d'une connexion anormale.

Configuration par Scripts sous Debian 6 et 7

Dans l'exemple suivant, expliquez le fonctionnement du script en détaillant les règles écrites :

```
#!/bin/bash
####################################
# proxy server IP
PROXY_SERVER="192.168.1.2"
# Interface connected to Internet
INTERNET="eth1"
# Interface connected to LAN
LAN_IN="eth0"
# Local Interface
LOCAL="lo"
# Squid port
PROXY_PORT="8080"
# DO NOT MODIFY BELOW
# Clean old firewall
```

```
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
# Load IPTABLES modules for NAT and IP conntrack support
modprobe ip_conntrack
modprobe ip_conntrack_ftp
# For win xp ftp client
modprobe ip_nat_ftp
echo 1 > /proc/sys/net/ipv4/ip_forward
# Setting default filter policy
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
# Unlimited access to loop back
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
# Allow UDP, DNS and Passive FTP
iptables -A INPUT -i $INTERNET -m state --state ESTABLISHED,RELATED -j ACCEPT
# set this system as a router for Rest of LAN
iptables --table nat --append POSTROUTING --out-interface $INTERNET -j MASQUERADE
iptables --append FORWARD --in-interface $LAN_IN -j ACCEPT
# unlimited access to LAN
iptables -A INPUT -i $LAN_IN -j ACCEPT
iptables -A OUTPUT -o $LAN_IN -j ACCEPT
# DNAT port 80 request comming from LAN systems to squid 3128 ($SQUID_PORT) aka transparent proxy
iptables -t nat -A PREROUTING -i $LAN_IN -p tcp --dport 80 -j DNAT --to $PROXY_SERVER:$PROXY_PORT
# iptables -t nat -A PREROUTING -i br0 -p tcp --dport 80 -j REDIRECT --to-port 3128
# if it is same system
iptables -t nat -A PREROUTING -i $INTERNET -p tcp --dport 80 -j REDIRECT --to-port $PROXY_PORT
# DROP everything and Log it
iptables -A INPUT -j LOG
iptables -A INPUT -j DROP
```

Dans l'exemple suivant, expliquez le fonctionnement du script en détaillant les règles écrites :

```
#!/bin/sh
#
# Generated iptables firewall script for the Linux 2.4 kernel
# Script generated by Easy Firewall Generator for IPTables 1.15
# copyright 2002 Timothy Scott Morizot
#
# Redhat chkconfig comments - firewall applied early,
#                             removed late
# chkconfig: 2345 08 92
# description: This script applies or removes iptables firewall rules
#
# This generator is primarily designed for RedHat installations,
# although it should be adaptable for others.
#
# It can be executed with the typical start and stop arguments.
# If used with stop, it will stop after flushing the firewall.
# The save and restore arguments will save or restore the rules
# from the /etc/sysconfig/iptables file.  The save and restore
# arguments are included to preserve compatibility with
# Redhat's or Fedora's init.d script if you prefer to use it.

# Redhat/Fedora installation instructions
#
# 1. Have the system link the iptables init.d startup script into run states
#    2, 3, and 5.
#    chkconfig --level 235 iptables on
#
# 2. Save this script and execute it to load the ruleset from this file.
#    You may need to run the dos2unix command on it to remove carraige returns.
#
# 3. To have it applied at startup, copy this script to
#    /etc/init.d/iptables.  It accepts stop, start, save, and restore
```

```
#    arguments.  (You may wish to save the existing one first.)
#    Alternatively, if you issue the 'service iptables save' command
#    the init.d script should save the rules and reload them at runtime.
#
# 4. For non-Redhat systems (or Redhat systems if you have a problem), you
#    may want to append the command to execute this script to rc.local.
#    rc.local is typically located in /etc and /etc/rc.d and is usually
#    the last thing executed on startup.  Simply add /path/to/script/script_name
#    on its own line in the rc.local file.

##############################################################################
#
# Local Settings
#

# sysctl location.  If set, it will use sysctl to adjust the kernel parameters.
# If this is set to the empty string (or is unset), the use of sysctl
# is disabled.

SYSCTL="/sbin/sysctl -w"

# To echo the value directly to the /proc file instead
# SYSCTL=""

# IPTables Location - adjust if needed

IPT="/sbin/iptables"
IPTS="/sbin/iptables-save"
IPTR="/sbin/iptables-restore"

# Internet Interface
INET_IFACE="eth1"

# Local Interface Information
```

```
LOCAL_IFACE="eth0"
LOCAL_IP="192.168.1.1"
LOCAL_NET="192.168.1.0/24"
LOCAL_BCAST="192.168.1.255"

# Localhost Interface

LO_IFACE="lo"
LO_IP="127.0.0.1"

# Save and Restore arguments handled here
if [ "$1" = "save" ]
then
    echo -n "Saving firewall to /etc/sysconfig/iptables ... "
    $IPTS > /etc/sysconfig/iptables
    echo "done"
    exit 0
elif [ "$1" = "restore" ]
then
    echo -n "Restoring firewall from /etc/sysconfig/iptables ... "
    $IPTR < /etc/sysconfig/iptables
    echo "done"
    exit 0
fi

##############################################################################
#
# Load Modules
#

echo "Loading kernel modules ..."

# You should uncomment the line below and run it the first time just to
# ensure all kernel module dependencies are OK.  There is no need to run
```

```
# every time, however.

# /sbin/depmod -a

# Unless you have kernel module auto-loading disabled, you should not
# need to manually load each of these modules.  Other than ip_tables,
# ip_conntrack, and some of the optional modules, I've left these
# commented by default.  Uncomment if you have any problems or if
# you have disabled module autoload.  Note that some modules must
# be loaded by another kernel module.

# core netfilter module
/sbin/modprobe ip_tables

# the stateful connection tracking module
/sbin/modprobe ip_conntrack

# filter table module
# /sbin/modprobe iptable_filter

# mangle table module
# /sbin/modprobe iptable_mangle

# nat table module
# /sbin/modprobe iptable_nat

# LOG target module
# /sbin/modprobe ipt_LOG

# This is used to limit the number of packets per sec/min/hr
# /sbin/modprobe ipt_limit

# masquerade target module
# /sbin/modprobe ipt_MASQUERADE
```

```
# filter using owner as part of the match
# /sbin/modprobe ipt_owner<WRAP center round important>
Importez une machine virtuelle vierge de CentOS 6 pour effectuer le LABS #8.
</WRAP>

# REJECT target drops the packet and returns an ICMP response.
# The response is configurable.  By default, connection refused.
# /sbin/modprobe ipt_REJECT

# This target allows packets to be marked in the mangle table
# /sbin/modprobe ipt_mark

# This target affects the TCP MSS
# /sbin/modprobe ipt_tcpmss

# This match allows multiple ports instead of a single port or range
# /sbin/modprobe multiport

# This match checks against the TCP flags
# /sbin/modprobe ipt_state

# This match catches packets with invalid flags
# /sbin/modprobe ipt_unclean

# The ftp nat module is required for non-PASV ftp support
/sbin/modprobe ip_nat_ftp

# the module for full ftp connection tracking
/sbin/modprobe ip_conntrack_ftp

# the module for full irc connection tracking
/sbin/modprobe ip_conntrack_irc
```

```
##############################################################################
#
# Kernel Parameter Configuration
#
# See http://ipsysctl-tutorial.frozentux.net/chunkyhtml/index.html
# for a detailed tutorial on sysctl and the various settings
# available.

# Required to enable IPv4 forwarding.
# Redhat users can try setting FORWARD_IPV4 in /etc/sysconfig/network to true
# Alternatively, it can be set in /etc/sysctl.conf
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/ip_forward
else
    $SYSCTL net.ipv4.ip_forward="1"
fi

# This enables dynamic address hacking.
# This may help if you have a dynamic IP address \(e.g. slip, ppp, dhcp\).
#if [ "$SYSCTL" = "" ]
#then
#    echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#else
#    $SYSCTL net.ipv4.ip_dynaddr="1"
#fi

# This enables SYN flood protection.
# The SYN cookies activation allows your system to accept an unlimited
# number of TCP connections while still trying to give reasonable
# service during a denial of service attack.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/tcp_syncookies
```

```
else
    $SYSCTL net.ipv4.tcp_syncookies="1"
fi

# This enables source validation by reversed path according to RFC1812.
# In other words, did the response packet originate from the same interface
# through which the source packet was sent?  It's recommended for single-homed
# systems and routers on stub networks.  Since those are the configurations
# this firewall is designed to support, I turn it on by default.
# Turn it off if you use multiple NICs connected to the same network.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
else
    $SYSCTL net.ipv4.conf.all.rp_filter="1"
fi

# This option allows a subnet to be firewalled with a single IP address.
# It's used to build a DMZ.  Since that's not a focus of this firewall
# script, it's not enabled by default, but is included for reference.
# See: http://www.sjdjweis.com/linux/proxyarp/
#if [ "$SYSCTL" = "" ]
#then
#    echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#else
#    $SYSCTL net.ipv4.conf.all.proxy_arp="1"
#fi

# The following kernel settings were suggested by Alex Weeks. Thanks!

# This kernel parameter instructs the kernel to ignore all ICMP
# echo requests sent to the broadcast address.  This prevents
# a number of smurfs and similar DoS nasty attacks.
if [ "$SYSCTL" = "" ]
```

```
then
    echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
else
    $SYSCTL net.ipv4.icmp_echo_ignore_broadcasts="1"
fi

# This option can be used to accept or refuse source routed
# packets.  It is usually on by default, but is generally
# considered a security risk.  This option turns it off.
if [ "$SYSCTL" = "" ]
then
    echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
else
    $SYSCTL net.ipv4.conf.all.accept_source_route="0"
fi

# This option can disable ICMP redirects.  ICMP redirects
# are generally considered a security risk and shouldn't be
# needed by most systems using this generator.
#if [ "$SYSCTL" = "" ]
#then
#    echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
#else
#    $SYSCTL net.ipv4.conf.all.accept_redirects="0"
#fi

# However, we'll ensure the secure_redirects option is on instead.
# This option accepts only from gateways in the default gateways list.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/conf/all/secure_redirects
else
    $SYSCTL net.ipv4.conf.all.secure_redirects="1"
fi
```

```
# This option logs packets from impossible addresses.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
else
    $SYSCTL net.ipv4.conf.all.log_martians="1"
fi



##############################################################################
#
# Flush Any Existing Rules or Chains
#

echo "Flushing Tables ..."

# Reset Default Policies
$IPT -P INPUT ACCEPT
$IPT -P FORWARD ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
$IPT -t mangle -P PREROUTING ACCEPT
$IPT -t mangle -P OUTPUT ACCEPT

# Flush all rules
$IPT -F
$IPT -t nat -F
$IPT -t mangle -F

# Erase all non-default chains
$IPT -X
$IPT -t nat -X
```

```
$IPT -t mangle -X

if [ "$1" = "stop" ]
then
    echo "Firewall completely flushed!  Now running with no firewall."
    exit 0
fi

########################################################################
#
# Rules Configuration
#

########################################################################
#
# Filter Table
#
########################################################################

# Set Policies

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

########################################################################
#
# User-Specified Chains
#
# Create user chains to reduce the number of rules each packet
# must traverse.

echo "Create and populate custom rule chains ..."
```

```
# Create a chain to filter INVALID packets

$IPT -N bad_packets

# Create another chain to filter bad tcp packets

$IPT -N bad_tcp_packets

# Create separate chains for icmp, tcp (incoming and outgoing),
# and incoming udp packets.

$IPT -N icmp_packets

# Used for UDP packets inbound from the Internet
$IPT -N udp_inbound

# Used to block outbound UDP services from internal network
# Default to allow all
$IPT -N udp_outbound

# Used to allow inbound services if desired
# Default fail except for established sessions
$IPT -N tcp_inbound

# Used to block outbound services from internal network
# Default to allow all
$IPT -N tcp_outbound

##############################################################################
#
# Populate User Chains
#

# bad_packets chain
```

```
#

# Drop packets received on the external interface
# claiming a source of the local network
$IPT -A bad_packets -p ALL -i $INET_IFACE -s $LOCAL_NET -j LOG \
    --log-prefix "fp=bad_packets:2 a=DROP "
$IPT -A bad_packets -p ALL -i $INET_IFACE -s $LOCAL_NET -j DROP

# Drop INVALID packets immediately
$IPT -A bad_packets -p ALL -m state --state INVALID -j LOG \
    --log-prefix "fp=bad_packets:1 a=DROP "

$IPT -A bad_packets -p ALL -m state --state INVALID -j DROP

# Then check the tcp packets for additional problems
$IPT -A bad_packets -p tcp -j bad_tcp_packets

# All good, so return
$IPT -A bad_packets -p ALL -j RETURN

# bad_tcp_packets chain
#
# All tcp packets will traverse this chain.
# Every new connection attempt should begin with
# a syn packet.  If it doesn't, it is likely a
# port scan.  This drops packets in state
# NEW that are not flagged as syn packets.

# Return to the calling chain if the bad packets originate
# from the local interface. This maintains the approach
# throughout this firewall of a largely trusted internal
# network.
$IPT -A bad_tcp_packets -p tcp -i $LOCAL_IFACE -j RETURN
```

```
# However, I originally did apply this filter to the forward chain
# for packets originating from the internal network.  While I have
# not conclusively determined its effect, it appears to have the
# interesting side effect of blocking some of the ad systems.
# Apparently some ad systems have the browser initiate a NEW
# connection that is not flagged as a syn packet to retrieve
# the ad image.  If you wish to experiment further comment the
# rule above. If you try it, you may also wish to uncomment the
# rule below.  It will keep those packets from being logged.
# There are a lot of them.
# $IPT -A bad_tcp_packets -p tcp -i $LOCAL_IFACE ! --syn -m state \
#     --state NEW -j DROP

$IPT -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
    --log-prefix "fp=bad_tcp_packets:1 a=DROP "
$IPT -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL NONE -j LOG \
    --log-prefix "fp=bad_tcp_packets:2 a=DROP "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL NONE -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL ALL -j LOG \
    --log-prefix "fp=bad_tcp_packets:3 a=DROP "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL ALL -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL FIN,URG,PSH -j LOG \
    --log-prefix "fp=bad_tcp_packets:4 a=DROP "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j LOG \
    --log-prefix "fp=bad_tcp_packets:5 a=DROP "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,RST SYN,RST -j LOG \
```

```
    --log-prefix "fp=bad_tcp_packets:6 a=DROP "
$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,RST SYN,RST -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,FIN SYN,FIN -j LOG \
    --log-prefix "fp=bad_tcp_packets:7 a=DROP "
$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

# All good, so return
$IPT -A bad_tcp_packets -p tcp -j RETURN

# icmp_packets chain
#
# This chain is for inbound (from the Internet) icmp packets only.
# Type 8 (Echo Request) is not accepted by default
# Enable it if you want remote hosts to be able to reach you.
# 11 (Time Exceeded) is the only one accepted
# that would not already be covered by the established
# connection rule.  Applied to INPUT on the external interface.
#
# See: http://www.ee.siue.edu/~rwalden/networking/icmp.html
# for more info on ICMP types.
#
# Note that the stateful settings allow replies to ICMP packets.
# These rules allow new packets of the specified types.

# ICMP packets should fit in a Layer 2 frame, thus they should
# never be fragmented.  Fragmented ICMP packets are a typical sign
# of a denial of service attack.
$IPT -A icmp_packets --fragment -p ICMP -j LOG \
    --log-prefix "fp=icmp_packets:1 a=DROP "
$IPT -A icmp_packets --fragment -p ICMP -j DROP

# Echo - uncomment to allow your system to be pinged.
# Uncomment the LOG command if you also want to log PING attempts
```

```
#
# $IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j LOG \
#    --log-prefix "fp=icmp_packets:2 a=ACCEPT "
# $IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT

# By default, however, drop pings without logging. Blaster
# and other worms have infected systems blasting pings.
# Comment the line below if you want pings logged, but it
# will likely fill your logs.
$IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j DROP

# Time Exceeded
$IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

# Not matched, so return so it will be logged
$IPT -A icmp_packets -p ICMP -j RETURN

# TCP & UDP
# Identify ports at:
#    http://www.chebucto.ns.ca/~rakerman/port-table.html
#    http://www.iana.org/assignments/port-numbers

# udp_inbound chain
#
# This chain describes the inbound UDP packets it will accept.
# It's applied to INPUT on the external or Internet interface.
# Note that the stateful settings allow replies.
# These rules are for new requests.
# It drops netbios packets (windows) immediately without logging.

# Drop netbios calls
# Please note that these rules do not really change the way the firewall
# treats netbios connections.  Connections from the localhost and
# internal interface (if one exists) are accepted by default.
```

```
# Responses from the Internet to requests initiated by or through
# the firewall are also accepted by default.  To get here, the
# packets would have to be part of a new request received by the
# Internet interface.  You would have to manually add rules to
# accept these.  I added these rules because some network connections,
# such as those via cable modems, tend to be filled with noise from
# unprotected Windows machines.  These rules drop those packets
# quickly and without logging them.  This prevents them from traversing
# the whole chain and keeps the log from getting cluttered with
# chatter from Windows systems.
$IPT -A udp_inbound -p UDP -s 0/0 --destination-port 137 -j DROP
$IPT -A udp_inbound -p UDP -s 0/0 --destination-port 138 -j DROP

# Ident requests (Port 113) must have a REJECT rule rather than the
# default DROP rule.  This is the minimum requirement to avoid
# long delays while connecting.  Also see the tcp_inbound rule.
$IPT -A udp_inbound -p UDP -s 0/0 --destination-port 113 -j REJECT

# A more sophisticated configuration could accept the ident requests.
# $IPT -A udp_inbound -p UDP -s 0/0 --destination-port 113 -j ACCEPT

# However, if this is a gateway system that masquerades/nats for internal systems
# and the internal systems wish to chat, a simple changing these rules to
# ACCEPT won't work.  The ident daemon on the gateway will need to know how
# to handle the requests.  The stock daemon in most linux distributions
# can't do that.   oidentd is one package that can.
# See: http://dev.ojnk.net/

# Network Time Protocol (NTP) Server
$IPT -A udp_inbound -p UDP -s 0/0 --destination-port 123 -j ACCEPT

# Dynamic Address
# If DHCP, the initial request is a broadcast. The response
# doesn't exactly match the outbound packet.  This explicitly
```

```
# allow the DHCP ports to alleviate this problem.
# If you receive your dynamic address by a different means, you
# can probably comment this line.
$IPT -A udp_inbound -p UDP -s 0/0 --source-port 67 --destination-port 68 \
     -j ACCEPT



# Not matched, so return for logging
$IPT -A udp_inbound -p UDP -j RETURN

# udp_outbound chain
#
# This chain is used with a private network to prevent forwarding for
# UDP requests on specific protocols.  Applied to the FORWARD rule from
# the internal network.  Ends with an ACCEPT



# No match, so ACCEPT
$IPT -A udp_outbound -p UDP -s 0/0 -j ACCEPT

# tcp_inbound chain
#
# This chain is used to allow inbound connections to the
# system/gateway.  Use with care.  It defaults to none.
# It's applied on INPUT from the external or Internet interface.

# Ident requests (Port 113) must have a REJECT rule rather than the
# default DROP rule.  This is the minimum requirement to avoid
# long delays while connecting.  Also see the tcp_inbound rule.
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 113 -j REJECT

# A more sophisticated configuration could accept the ident requests.
# $IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 113 -j ACCEPT
```

```
# However, if this is a gateway system that masquerades/nats for internal systems
# and the internal systems wish to chat, a simple changing these rules to
# ACCEPT won't work.  The ident daemon on the gateway will need to know how
# to handle the requests.  The stock daemon in most linux distributions
# can't do that.   oidentd is one package that can.
# See: http://dev.ojnk.net/

# Web Server

# HTTP
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 80 -j ACCEPT

# HTTPS (Secure Web Server)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 443 -j ACCEPT

# FTP Server (Control)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 21 -j ACCEPT

# FTP Client (Data Port for non-PASV transfers)
$IPT -A tcp_inbound -p TCP -s 0/0 --source-port 20 -j ACCEPT

# Passive FTP
#
# With passive FTP, the server provides a port to the client
# and allows the client to initiate the connection rather
# than initiating the connection with the client from the data port.
# Web browsers and clients operating behind a firewall generally
# use passive ftp transfers.  A general purpose FTP server
# will need to support them.
#
# However, by default an FTP server will select a port from the entire
# range of high ports.  It is not particularly safe to open all
# high ports.  Fortunately, that range can be restricted.  This
# firewall presumes that the range has been restricted to a specific
```

```
# selected range.  That range must also be configured in the ftp server.
#
# Instructions for specifying the port range for the wu-ftpd server
# can be found here:
# http://www.wu-ftpd.org/man/ftpaccess.html
# (See the passive ports option.)
#
# Instructions for the ProFTPD server can be found here:
# http://proftpd.linux.co.uk/localsite/Userguide/linked/x861.html

$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 62000:64000 -j ACCEPT

# Email Server (SMTP)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 25 -j ACCEPT

# Email Server (POP3)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 110 -j ACCEPT

# Email Server (IMAP4)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 143 -j ACCEPT

# SSL Email Server (POP3)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 995 -j ACCEPT

# SSL Email Server (IMAP4)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 993 -j ACCEPT

# sshd
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 22 -j ACCEPT

# ICQ File Transfers & Other Advanced Features
#
# ICQ supports a number of options beyond simple instant messaging.
# For those to function, the instant messaging system must allow
```

```
# new connections initiated from remote systems. This option will
# open a specified port range on the firewalled system.  The ICQ client
# on the firewalled system must also be configured to use the specified
# port range.

$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 5000:5100 -j ACCEPT

# MSN Messenger File Transfers
#
# Messenger supports file transfers.  For transfers initiated by
# remote systems to function, the system must allow
# new connections initiated from remote systems a specific port range.
# This option defaults to the port range 6891 through 6900.
# Unless the MSN Messenger client can be configured to specify any
# port range, don't change the default.

$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 6891:6900 -j ACCEPT

# User specified allowed UDP protocol
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 32500:36000 -j ACCEPT



# Not matched, so return so it will be logged
$IPT -A tcp_inbound -p TCP -j RETURN

# tcp_outbound chain
#
# This chain is used with a private network to prevent forwarding for
# requests on specific protocols.  Applied to the FORWARD rule from
# the internal network.  Ends with an ACCEPT



# No match, so ACCEPT
$IPT -A tcp_outbound -p TCP -s 0/0 -j ACCEPT
```

```
#############################################################################
#
# INPUT Chain
#

echo "Process INPUT chain ..."

# Allow all on localhost interface
$IPT -A INPUT -p ALL -i $LO_IFACE -j ACCEPT

# Drop bad packets
$IPT -A INPUT -p ALL -j bad_packets

# DOCSIS compliant cable modems
# Some DOCSIS compliant cable modems send IGMP multicasts to find
# connected PCs.  The multicast packets have the destination address
# 224.0.0.1.  You can accept them.  If you choose to do so,
# Uncomment the rule to ACCEPT them and comment the rule to DROP
# them  The firewall will drop them here by default to avoid
# cluttering the log.  The firewall will drop all multicasts
# to the entire subnet (224.0.0.1) by default.  To only affect
# IGMP multicasts, change '-p ALL' to '-p 2'.  Of course,
# if they aren't accepted elsewhere, it will only ensure that
# multicasts on other protocols are logged.
# Drop them without logging.
$IPT -A INPUT -p ALL -d 224.0.0.1 -j DROP
# The rule to accept the packets.
# $IPT -A INPUT -p ALL -d 224.0.0.1 -j ACCEPT

# Rules for the private network (accessing gateway system itself)
$IPT -A INPUT -p ALL -i $LOCAL_IFACE -s $LOCAL_NET -j ACCEPT
$IPT -A INPUT -p ALL -i $LOCAL_IFACE -d $LOCAL_BCAST -j ACCEPT
```

```
# Inbound Internet Packet Rules

# Accept Established Connections
$IPT -A INPUT -p ALL -i $INET_IFACE -m state --state ESTABLISHED,RELATED \
     -j ACCEPT

# Route the rest to the appropriate user chain
$IPT -A INPUT -p TCP -i $INET_IFACE -j tcp_inbound
$IPT -A INPUT -p UDP -i $INET_IFACE -j udp_inbound
$IPT -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

# Drop without logging broadcasts that get this far.
# Cuts down on log clutter.
# Comment this line if testing new rules that impact
# broadcast protocols.
$IPT -A INPUT -m pkttype --pkt-type broadcast -j DROP

# Log packets that still don't match
$IPT -A INPUT -j LOG --log-prefix "fp=INPUT:99 a=DROP "

##############################################################################
#
# FORWARD Chain
#

echo "Process FORWARD chain ..."

# Used if forwarding for a private network

# Drop bad packets
$IPT -A FORWARD -p ALL -j bad_packets

# Accept TCP packets we want to forward from internal sources
$IPT -A FORWARD -p tcp -i $LOCAL_IFACE -j tcp_outbound
```

```
# Accept UDP packets we want to forward from internal sources
$IPT -A FORWARD -p udp -i $LOCAL_IFACE -j udp_outbound

# If not blocked, accept any other packets from the internal interface
$IPT -A FORWARD -p ALL -i $LOCAL_IFACE -j ACCEPT

# Deal with responses from the internet
$IPT -A FORWARD -i $INET_IFACE -m state --state ESTABLISHED,RELATED \
     -j ACCEPT

# Port Forwarding is enabled, so accept forwarded traffic
$IPT -A FORWARD -p tcp -i $INET_IFACE --destination-port 8080 \
     --destination 192.168.1.50 -j ACCEPT

# Log packets that still don't match
$IPT -A FORWARD -j LOG --log-prefix "fp=FORWARD:99 a=DROP "

###############################################################################
#
# OUTPUT Chain
#

echo "Process OUTPUT chain ..."

# Generally trust the firewall on output

# However, invalid icmp packets need to be dropped
# to prevent a possible exploit.
$IPT -A OUTPUT -m state -p icmp --state INVALID -j DROP

# Localhost
$IPT -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPT -A OUTPUT -p ALL -o $LO_IFACE -j ACCEPT
```

```
# To internal network
$IPT -A OUTPUT -p ALL -s $LOCAL_IP -j ACCEPT
$IPT -A OUTPUT -p ALL -o $LOCAL_IFACE -j ACCEPT

# To internet
$IPT -A OUTPUT -p ALL -o $INET_IFACE -j ACCEPT

# Log packets that still don't match
$IPT -A OUTPUT -j LOG --log-prefix "fp=OUTPUT:99 a=DROP "

##############################################################################
#
# nat table
#
##############################################################################

# The nat table is where network address translation occurs if there
# is a private network.  If the gateway is connected to the Internet
# with a static IP, snat is used.  If the gateway has a dynamic address,
# masquerade must be used instead.  There is more overhead associated
# with masquerade, so snat is better when it can be used.
# The nat table has a builtin chain, PREROUTING, for dnat and redirects.
# Another, POSTROUTING, handles snat and masquerade.

echo "Load rules for nat table ..."

##############################################################################
#
# PREROUTING chain
#

# Port Forwarding
#
# Port forwarding forwards all traffic on a port or ports from
```

```
# the firewall to a computer on the internal LAN.  This can
# be required to support special situations.  For instance,
# this is the only way to support file transfers with an ICQ
# client on an internal computer.  It's also required if an internal
# system hosts a service such as a web server.  However, it's also
# a dangerous option.  It allows Internet computers access to
# your internal network.  Use it carefully and only if you're
# certain you know what you're doing.

$IPT -t nat -A PREROUTING -p tcp -i $INET_IFACE --destination-port 80:80 \
     -j DNAT --to-destination 192.168.1.50:8080

# This is a sample that will exempt a specific host from the transparent proxy
#$IPT -t nat -A PREROUTING -p tcp -s 192.168.1.50 --destination-port 80 \
#     -j RETURN
#$IPT -t nat -A PREROUTING -p tcp -s 192.168.1.50 --destination-port 443 \
#     -j RETURN

# Redirect HTTP for a transparent proxy
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 \
     -j REDIRECT --to-ports 3128
# Redirect HTTPS for a transparent proxy - commented by default
# $IPT -t nat -A PREROUTING -p tcp --destination-port 443 \
#     -j REDIRECT --to-ports 3128

################################################################################
#
# POSTROUTING chain
#

$IPT -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE

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

```
# mangle table
#
################################################################################

# The mangle table is used to alter packets.  It can alter or mangle them in
# several ways.  For the purposes of this generator, we only use its ability
# to alter the TTL in packets.  However, it can be used to set netfilter
# mark values on specific packets.  Those marks could then be used in another
# table like filter, to limit activities associated with a specific host, for
# instance.  The TOS target can be used to set the Type of Service field in
# the IP header.  Note that the TTL target might not be included in the
# distribution on your system.  If it is not and you require it, you will
# have to add it.  That may require that you build from source.

echo "Load rules for mangle table ..."

# Set the TTL in outbound packets to the same consistent value.
# A value around 128 is a good value.  Do not set this too high as
# it will adversely affect your network.  It is also considered bad
# form on the Internet.
$IPT -t mangle -A OUTPUT -o $INET_IFACE -j TTL --ttl-set 128
```

Utilisez la commande **system-config-firewall** pour lancer l'outil graphique **Configuration du pare-feu** et constatez la configuration de netfilter.

La Configuration par firewalld sous Debian 8

Importez une machine virtuelle vierge de Debian 8.

firewalld est à Netfilter ce que NetworkManager est au réseau. firewalld utilise des **zones** - des jeux de règles pré-définis dans lesquels sont placés les interfaces :

- **trusted** - un réseau fiable. Dans ce cas tous les ports sont autorisés,
- **work**, **home**, **internal** - un réseau partiellement fiable. Dans ce cas quelques ports sont autorisés,
- **dmz**, **public**, **external** - un réseau non fiable. Dans ce cas peu de ports sont autorisés,
- **block**, **drop** - tout est interdit. La zone drop n'envoie pas de messages d'erreurs.

> Une interface ne peut être que dans une zone à la fois tandis que plusieurs interfaces peuvent être dans la même zone.

Sous Debian 8, firewalld n'est pas installé par défaut :

```
root@debian8:~# apt-get install firewalld
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libevent-core-2.0-5 libevent-pthreads-2.0-5
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  ebtables python-decorator python-slip python-slip-dbus
The following NEW packages will be installed:
  ebtables firewalld python-decorator python-slip python-slip-dbus
0 upgraded, 5 newly installed, 0 to remove and 1 not upgraded.
Need to get 436 kB of archives.
After this operation, 2,950 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Le service firewalld est déjà lancé et activé :

```
root@debian8:~# systemctl status firewalld.service
```

```
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/lib/systemd/system/firewalld.service; enabled)
   Active: active (running) since Wed 2017-07-26 09:10:38 BST; 24s ago
 Main PID: 8093 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─8093 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

Jul 26 09:10:38 debian8 systemd[1]: Started firewalld - dynamic firewall daemon.
```

La Configuration de Base de firewalld

La configuration par défaut de firewalld se trouve dans **/usr/lib/firewalld** :

```
root@debian8:~# ls -lR /usr/lib/firewalld/
/usr/lib/firewalld/:
total 12
drwxr-xr-x 2 root root 4096 Jul 26 09:10 icmptypes
drwxr-xr-x 2 root root 4096 Jul 26 09:10 services
drwxr-xr-x 2 root root 4096 Jul 26 09:10 zones

/usr/lib/firewalld/icmptypes:
total 36
-rw-r--r-- 1 root root 222 Oct 16  2014 destination-unreachable.xml
-rw-r--r-- 1 root root 173 Oct 16  2014 echo-reply.xml
-rw-r--r-- 1 root root 210 Oct 16  2014 echo-request.xml
-rw-r--r-- 1 root root 225 Oct 16  2014 parameter-problem.xml
-rw-r--r-- 1 root root 185 Oct 16  2014 redirect.xml
-rw-r--r-- 1 root root 227 Oct 16  2014 router-advertisement.xml
-rw-r--r-- 1 root root 223 Oct 16  2014 router-solicitation.xml
-rw-r--r-- 1 root root 248 Oct 16  2014 source-quench.xml
-rw-r--r-- 1 root root 253 Oct 16  2014 time-exceeded.xml

/usr/lib/firewalld/services:
```

```
total 248
-rw-r--r-- 1 root root 412 Oct 16  2014 amanda-client.xml
-rw-r--r-- 1 root root 447 Oct 16  2014 amanda-k5-client.xml
-rw-r--r-- 1 root root 320 Oct 16  2014 bacula-client.xml
-rw-r--r-- 1 root root 346 Oct 16  2014 bacula.xml
-rw-r--r-- 1 root root 305 Oct 16  2014 dhcpv6-client.xml
-rw-r--r-- 1 root root 234 Oct 16  2014 dhcpv6.xml
-rw-r--r-- 1 root root 227 Oct 16  2014 dhcp.xml
-rw-r--r-- 1 root root 346 Oct 16  2014 dns.xml
-rw-r--r-- 1 root root 836 Oct 16  2014 freeipa-ldaps.xml
-rw-r--r-- 1 root root 836 Oct 16  2014 freeipa-ldap.xml
-rw-r--r-- 1 root root 315 Oct 16  2014 freeipa-replication.xml
-rw-r--r-- 1 root root 374 Oct 16  2014 ftp.xml
-rw-r--r-- 1 root root 476 Oct 16  2014 high-availability.xml
-rw-r--r-- 1 root root 448 Oct 16  2014 https.xml
-rw-r--r-- 1 root root 353 Oct 16  2014 http.xml
-rw-r--r-- 1 root root 372 Oct 16  2014 imaps.xml
-rw-r--r-- 1 root root 454 Oct 16  2014 ipp-client.xml
-rw-r--r-- 1 root root 427 Oct 16  2014 ipp.xml
-rw-r--r-- 1 root root 517 Oct 16  2014 ipsec.xml
-rw-r--r-- 1 root root 182 Oct 16  2014 kadmin.xml
-rw-r--r-- 1 root root 233 Oct 16  2014 kerberos.xml
-rw-r--r-- 1 root root 221 Oct 16  2014 kpasswd.xml
-rw-r--r-- 1 root root 232 Oct 16  2014 ldaps.xml
-rw-r--r-- 1 root root 199 Oct 16  2014 ldap.xml
-rw-r--r-- 1 root root 385 Oct 16  2014 libvirt-tls.xml
-rw-r--r-- 1 root root 389 Oct 16  2014 libvirt.xml
-rw-r--r-- 1 root root 424 Oct 16  2014 mdns.xml
-rw-r--r-- 1 root root 211 Oct 16  2014 mountd.xml
-rw-r--r-- 1 root root 190 Oct 16  2014 ms-wbt.xml
-rw-r--r-- 1 root root 171 Oct 16  2014 mysql.xml
-rw-r--r-- 1 root root 324 Oct 16  2014 nfs.xml
-rw-r--r-- 1 root root 389 Oct 16  2014 ntp.xml
-rw-r--r-- 1 root root 335 Oct 16  2014 openvpn.xml
```

```
-rw-r--r-- 1 root root 433 Oct 16  2014 pmcd.xml
-rw-r--r-- 1 root root 474 Oct 16  2014 pmproxy.xml
-rw-r--r-- 1 root root 544 Oct 16  2014 pmwebapis.xml
-rw-r--r-- 1 root root 460 Oct 16  2014 pmwebapi.xml
-rw-r--r-- 1 root root 357 Oct 16  2014 pop3s.xml
-rw-r--r-- 1 root root 181 Oct 16  2014 postgresql.xml
-rw-r--r-- 1 root root 509 Oct 16  2014 privoxy.xml
-rw-r--r-- 1 root root 261 Oct 16  2014 proxy-dhcp.xml
-rw-r--r-- 1 root root 297 Oct 16  2014 puppetmaster.xml
-rw-r--r-- 1 root root 446 Oct 16  2014 radius.xml
-rw-r--r-- 1 root root 214 Oct 16  2014 rpc-bind.xml
-rw-r--r-- 1 root root 384 Oct 16  2014 samba-client.xml
-rw-r--r-- 1 root root 461 Oct 16  2014 samba.xml
-rw-r--r-- 1 root root 337 Oct 16  2014 sane.xml
-rw-r--r-- 1 root root 550 Oct 16  2014 smtp.xml
-rw-r--r-- 1 root root 173 Oct 16  2014 squid.xml
-rw-r--r-- 1 root root 463 Oct 16  2014 ssh.xml
-rw-r--r-- 1 root root 496 Oct 16  2014 synergy.xml
-rw-r--r-- 1 root root 393 Oct 16  2014 telnet.xml
-rw-r--r-- 1 root root 301 Oct 16  2014 tftp-client.xml
-rw-r--r-- 1 root root 437 Oct 16  2014 tftp.xml
-rw-r--r-- 1 root root 771 Oct 16  2014 tor-socks.xml
-rw-r--r-- 1 root root 211 Oct 16  2014 transmission-client.xml
-rw-r--r-- 1 root root 475 Oct 16  2014 vnc-server.xml
-rw-r--r-- 1 root root 310 Oct 16  2014 wbem-https.xml
-rw-r--r-- 1 root root 509 Oct 16  2014 xmpp-bosh.xml
-rw-r--r-- 1 root root 488 Oct 16  2014 xmpp-client.xml
-rw-r--r-- 1 root root 264 Oct 16  2014 xmpp-local.xml
-rw-r--r-- 1 root root 545 Oct 16  2014 xmpp-server.xml

/usr/lib/firewalld/zones:
total 36
-rw-r--r-- 1 root root 299 Oct 16  2014 block.xml
-rw-r--r-- 1 root root 293 Oct 16  2014 dmz.xml
```

```
-rw-r--r-- 1 root root 291 Oct 16  2014 drop.xml
-rw-r--r-- 1 root root 304 Oct 16  2014 external.xml
-rw-r--r-- 1 root root 369 Oct 16  2014 home.xml
-rw-r--r-- 1 root root 384 Oct 16  2014 internal.xml
-rw-r--r-- 1 root root 315 Oct 16  2014 public.xml
-rw-r--r-- 1 root root 162 Oct 16  2014 trusted.xml
-rw-r--r-- 1 root root 311 Oct 16  2014 work.xml
```

Ces fichiers sont au format **xml**, par exemple :

```
root@debian8:~# cat /usr/lib/firewalld/zones/home.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Home</short>
  <description>For use in home areas. You mostly trust the other computers on networks to not harm your computer.
Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="mdns"/>
  <service name="samba-client"/>
  <service name="dhcpv6-client"/>
</zone>
```

La configuration de firewalld ainsi que les définitions et règles personnalisées se trouvent dans **/etc/firewalld** :

```
root@debian8:~# ls -lR /etc/firewalld/
/etc/firewalld/:
total 20
-rw-r--r-- 1 root root 1026 Oct 16  2014 firewalld.conf
drwxr-xr-x 2 root root 4096 Oct 16  2014 icmptypes
-rw-r--r-- 1 root root  271 Oct 16  2014 lockdown-whitelist.xml
drwxr-xr-x 2 root root 4096 Oct 16  2014 services
drwxr-xr-x 2 root root 4096 Oct 16  2014 zones

/etc/firewalld/icmptypes:
```

```
total 0

/etc/firewalld/services:
total 0

/etc/firewalld/zones:
total 0
```

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

```
root@debian8:~# cat /etc/firewalld/firewalld.conf
# firewalld config file

# default zone
# The default zone used if an empty zone string is used.
# Default: public
DefaultZone=public

# Minimal mark
# Marks up to this minimum are free for use for example in the direct
# interface. If more free marks are needed, increase the minimum
# Default: 100
MinimalMark=100

# Clean up on exit
# If set to no or false the firewall configuration will not get cleaned up
# on exit or stop of firewalld
# Default: yes
CleanupOnExit=yes

# Lockdown
# If set to enabled, firewall changes with the D-Bus interface will be limited
# to applications that are listed in the lockdown whitelist.
# The lockdown whitelist file is lockdown-whitelist.xml
```

```
# Default: no
Lockdown=no

# IPv6_rpfilter
# Performs a reverse path filter test on a packet for IPv6. If a reply to the
# packet would be sent via the same interface that the packet arrived on, the
# packet will match and be accepted, otherwise dropped.
# The rp_filter for IPv4 is controlled using sysctl.
# Default: yes
IPv6_rpfilter=yes
```

La Commande firewall-cmd

firewalld s'appuie sur netfilter. Pour cette raison, l'utilisation de firewall-cmd est incompatible avec l'utilisation des commandes iptables et system-config-firewall.

> firewall-cmd est le front-end de firewalld en ligne de commande. Il existe aussi la commande **firewall-config** qui lance un outil de configuration graphique.

Pour obtenir la liste de toutes les zones prédéfinies, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

Pour obtenir la liste de toutes les services prédéfinis, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --get-services
amanda-client amanda-k5-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns freeipa-ldap freeipa-ldaps
freeipa-replication ftp high-availability http https imaps ipp ipp-client ipsec kadmin kerberos kpasswd ldap
ldaps libvirt libvirt-tls mdns mountd ms-wbt mysql nfs ntp openvpn pmcd pmproxy pmwebapi pmwebapis pop3s
```

```
postgresql privoxy proxy-dhcp puppetmaster radius rpc-bind samba samba-client sane smtp squid ssh synergy telnet
tftp tftp-client tor-socks transmission-client vnc-server wbem-https xmpp-bosh xmpp-client xmpp-local xmpp-server
```

Pour obtenir la liste de toutes les types ICMP prédéfinis, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --get-icmptypes
destination-unreachable echo-reply echo-request parameter-problem redirect router-advertisement router-
solicitation source-quench time-exceeded
```

Pour obtenir la liste des zones de la configuration courante, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --get-active-zones
public
  interfaces: eth0
```

Pour obtenir la liste des zones de la configuration courante pour une interface spécifique, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --get-zone-of-interface=eth0
public
```

Pour obtenir la liste des services autorisés pour la zone public, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=public --list-services
dhcpv6-client ssh
```

Pour obtenir toute la configuration pour la zone public, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=public --list-all
public (default, active)
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
```

```
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Pour obtenir la liste complète de toutes les zones et leurs configurations, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --list-all-zones
block
  interfaces:
  sources:
  services:
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
dmz
  interfaces:
  sources:
  services: ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
drop
  interfaces:
  sources:
  services:
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
```

```
  rich rules:
external
  interfaces:
  sources:
  services: ssh
  ports:
  masquerade: yes
  forward-ports:
  icmp-blocks:
  rich rules:
home
  interfaces:
  sources:
  services: dhcpv6-client mdns samba-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
internal
  interfaces:
  sources:
  services: dhcpv6-client mdns samba-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
public (default, active)
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
```

```
  forward-ports:
  icmp-blocks:
  rich rules:
trusted
  interfaces:
  sources:
  services:
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
work
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Pour changer la zone par défaut de public à work, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --set-default-zone=work
success
root@debian8:~# firewall-cmd --get-active-zones
work
  interfaces: eth0
```

HERE

Pour ajouter l'interface ip_fixe à la zone work, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --add-interface=ip_fixe
success
root@debian8:~# firewall-cmd --get-active-zones
work
  interfaces: eth0 ip_fixe
```

Pour supprimer l'interface ip_fixe à la zone work, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --remove-interface=ip_fixe
success
root@debian8:~# firewall-cmd --get-active-zones
work
  interfaces: eth0
```

Pour ajouter le service **http** à la zone **work**, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --add-service=http
success
root@debian8:~# firewall-cmd --zone=work --list-services
dhcpv6-client http ssh
```

Pour supprimer le service **http** de la zone **work**, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --remove-service=http
success
root@debian8:~# firewall-cmd --zone=work --list-services
dhcpv6-client ssh
```

Pour ajouter un nouveau bloc ICMP, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --add-icmp-block=echo-reply
success
root@debian8:~# firewall-cmd --zone=work --list-icmp-blocks
```

```
echo-reply
```

Pour supprimer un bloc ICMP, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --remove-icmp-block=echo-reply
success
root@debian8:~# firewall-cmd --zone=work --list-icmp-blocks
root@debian8:~#
```

Pour ajouter le port 591/tcp à la zone work, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --add-port=591/tcp
success
root@debian8:~# firewall-cmd --zone=work --list-ports
591/tcp
```

Pour supprimer le port 591/tcp à la zone work, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --zone=work --remove-port=591/tcp
success
root@debian8:~# firewall-cmd --zone=work --list-ports
root@debian8:~#
```

Pour créer un nouveau service, il convient de :

- copier un fichier existant se trouvant dans le répertoire **/usr/lib/firewalld/services** vers **/etc/firewalld/services**,
- modifier le fichier,
- recharger la configuration de firewalld,
- vérifier que firewalld voit le nouveau service.

Par exemple :

```
root@debian8:~# cp /usr/lib/firewalld/services/http.xml /etc/firewalld/services/filemaker.xml
root@debian8:~# cat /etc/firewalld/services/filemaker.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>WWW (HTTP)</short>
  <description>HTTP is the protocol used to serve Web pages. If you plan to make your Web server publicly
available, enable this option. This option is not required for viewing pages locally or developing Web
pages.</description>
  <port protocol="tcp" port="80"/>
</service>
root@debian8:~# vi /etc/firewalld/services/filemaker.xml
root@debian8:~# cat /etc/firewalld/services/filemaker.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>FileMakerPro</short>
  <description>fichier de service firewalld pour FileMaker Pro</description>
  <port protocol="tcp" port="591"/>
</service>
root@debian8:~# firewall-cmd --reload
success
root@debian8:~# firewall-cmd --get-services
amanda-client amanda-k5-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns filemaker freeipa-ldap freeipa-
ldaps freeipa-replication ftp high-availability http https imaps ipp ipp-client ipsec kadmin kerberos kpasswd
ldap ldaps libvirt libvirt-tls mdns mountd ms-wbt mysql nfs ntp openvpn pmcd pmproxy pmwebapi pmwebapis pop3s
postgresql privoxy proxy-dhcp puppetmaster radius rpc-bind samba samba-client sane smtp squid ssh synergy telnet
tftp tftp-client tor-socks transmission-client vnc-server wbem-https xmpp-bosh xmpp-client xmpp-local xmpp-server
```

La Configuration Avancée de firewalld

La configuration de base de firewalld ne permet que la configuration des zones, services, blocs ICMP et les ports non-standard. Cependant firewalld peut également être configuré avec des **Rich Rules** ou ***Règles Riches***. Rich Rules ou Règles Riches évaluent des **critères** pour ensuite entreprendre une **action**.

Les **Critères** sont :

- **source address=“<adresse_IP>“**
- **destination address=”<adresse_IP>“**,
- **rule port port=”<numéro_du_port>“**,
- **service name=<nom_d'un_sevice_prédéfini>**.

Les **Actions** sont :

- **accept**,
- **reject**,
  - une Action reject peut être associée avec un message d'erreur spécifique par la clause **type=”<type_d'erreur>**,
- **drop**.

Saisissez la commande suivante pour ouvrir le port 80 :

```
root@debian8:~# firewall-cmd --add-rich-rule='rule port port="80" protocol="tcp" accept'
success
```

Notez que la Rich Rule doit être entourée de caractères '.

Saisissez la commande suivante pour visualiser la règle iptables pour IPv4 :

```
root@debian8:~# iptables -L -n | grep 80
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 ctstate NEW
```

Saisissez la commande suivante pour visualiser la règle iptables pour IPv6 :

```
root@debian8:~# ip6tables -L -n | grep 80
ACCEPT     udp      ::/0                 fe80::/64           udp dpt:546 ctstate NEW
ACCEPT     tcp      ::/0                 ::/0                tcp dpt:80 ctstate NEW
```

Important - Notez que la Rich Rule a créé deux règles, une pour IPv4 et une deuxième pour IPv6. Une règle peut être créée pour IPv4 seul en incluant le Critère **family=ipv4**. De la même façon, une règle peut être créée pour IPv6 seul en incluant le Critère **family=ipv6**.

Cette nouvelle règle est écrite en mémoire mais non pas sur disque. Pour l'écrire sur disque dans le fichier zone se trouvant dans **/etc/firewalld**, il faut ajouter l'option **–permanent** :

```
root@debian8:~# firewall-cmd --add-rich-rule='rule port port="80" protocol="tcp" accept' --permanent
success
root@debian8:~# cat /etc/firewalld/zones/work.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Work</short>
  <description>For use in work areas. You mostly trust the other computers on networks to not harm your computer.
Only selected incoming connections are accepted.</description>
  <service name="dhcpv6-client"/>
  <service name="ssh"/>
  <rule>
    <port protocol="tcp" port="80"/>
    <accept/>
  </rule>
</zone>
```

Important - Attention ! La règle ajoutée avec l'option –permanent n'est pas prise en compte imédiatement mais uniquement au prochain redémmarge. Pour qu'une règle soit appliquée immédiatement **et** être écrite sur disque, il faut saisir la commande deux fois dont une avec l'option –permanent et l'autre sans l'option –permanent.

Pour visualiser cette règle dans la configuration de firewalld, il convient de saisir la commande suivante :

```
root@debian8:~# firewall-cmd --list-all-zones
...
work (default, active)
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
    rule port port="80" protocol="tcp" accept
```

Notez que la Rich Rule est créée dans la Zone par Défaut. Il est possible de créer une Rich Rule dans une autre zone en utilisant l'option **–zone=<zone>** de la commande firewall-cmd :

```
root@debian8:~# firewall-cmd --zone=public --add-rich-rule='rule port port="80" protocol="tcp" accept'
success
root@debian8:~# firewall-cmd --list-all-zones
...
public
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
    rule port port="80" protocol="tcp" accept
trusted
  interfaces:
```

```
  sources:
  services:
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
work (default, active)
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
    rule port port="80" protocol="tcp" accept
```

Pour supprimer une Rich Rule, il faut copier la ligne entière la concernant qui se trouve dans la sortie de la commande **firewall-cmd –list-all-zones** :

```
root@debian8:~# firewall-cmd --zone=public --remove-rich-rule='rule port port="80" protocol="tcp" accept'
success
```

Le mode Panic de firewalld

Le mode Panic de firewalld permet de bloquer tout le trafic avec une seule commande. Pour connaître l'état du mode Panic, utilisez la commande suivante :

```
root@debian8:~# firewall-cmd --query-panic
no
```

Pour activer le mode Panic, il convient de saisir la commande suivante :

```
root@debian8:~# firewall-cmd --panic-on
success
root@debian8:~# firewall-cmd --query-panic
yes
```

Pour désactiver le mode Panic, il convient de saisir la commande suivante :

```
root@debian8:~# firewall-cmd --panic-off
success
root@debian8:~# firewall-cmd --query-panic
no
```

Annexe #1 - Comprendre les Réseaux

Présentation des Réseaux

La définition d'un réseau peut être résumé ainsi :

- un ensemble d'**Equipements** (systèmes et périphériques) communiquant entre eux,
- une entité destinée au transport de données dans différents environnements.

Pour que la communication soit efficace, elle doit respecter les critères suivants :

- présenter des informations compréhensibles par tous les participants,
- être compatible avec un maximum d'interlocuteurs différents (dans le cas d'un réseau, les interlocuteurs sont des équipements : imprimantes, ordinateurs, clients, serveurs, téléphones...),
- si l'interlocuteur n'est pas disponible, les informations ne doivent pas se perdre,
- permettre une réduction des coûts (par ex. interconnexion à bas coût),
- permettre une productivité accrue (par ex. interconnexion à haut débit),
- être sécurisée si les informations à transmettre sont dites sensibles,
- garantir l'**unicité** et de l'**universalité** de l'**accès à l'information**.

On peut distinguer deux familles d'**Equipements** - les **Eléments Passifs** et les **Eléments Actifs**.

Les **Eléments Passifs** transmettent le signal d'un point à un autre :

- **Les Infrastructures ou Supports** - des câbles, de l'atmosphère ou des fibres optiques permettant de relier **physiquement** des équipements,
- **La Topologie** - l'architecture d'un réseau définissant les connexions entre les **Equipements** et, éventuellement, la hiérarchie entre eux.

Les **Eléments Actifs** sont des équipements qui consomment de l'énergie en traitant ou en interprétant le signal. Les **Equipements** sont classés selon leurs fonctions :

- **Equipement de Distribution Interne au Réseau** - Répartiteur (Hub, Switch, Commutateur etc.), Borne d'accès (Hotspot), Convertisseur de signal (Transciever), Amplificateur (Répéteur) ...,
- **Equipement d'Interconnexion de Réseaux** - Routeurs, Ponts ...,
- **Nœuds** et **Interfaces Réseaux** - postes informatiques, équipements en réseau ....

Un **Nœud** est une extrémité de connexion qui peut être une intersection de plusieurs connexions ou de plusieurs **Equipements**.

Une **Interface Réseau** est une prise ou élément d'un **Equipement Actif** faisant la connexion vers d'autres **Equipements** réseaux et qui reçoit et émet des données.

> Dans le cas d'un mélange d'**Equipements** non-homogènes en termes de performances au sein du même réseau, c'est la loi du plus faible qui emporte.

Tous les **Equipements** connectés au même support doivent respecter un ensemble de règles appelé une **Protocole de Communication**.

Les **Protocoles de Communication** définissent de façon formelle et interopérable la manière dont les informations sont échangées entre les **Equipements**.

Des **Logiciels**, dédiés à la gestion de ces **Protocoles de Communication**, sont installés sur des **Equipements d'Interconnexion** afin de fournir des fonctions de contrôle permettant une communication entre les **Equipements**.

Se basant sur des **Protocoles de Communication**, des **Services** fournissent des fonctionnalités accessibles aux utilisateurs ou d'autres programmes.

L'ensemble des **Equipements**, **Logiciels** et **Protocoles de Communication** constitue l'**Architecture Réseau**.

Classification des Réseaux

Les réseaux peuvent être classifiés de trois façon différentes :

- par **Mode de Transmission**,
- par **Topologie**,
- par **Étendue**.

Classification par Mode de Transmission

Il existe deux **Classes** de réseaux dans cette classification :

- les **Réseaux en Mode de Diffusion**,
  - utilise un seul support de transmission,
  - le message est envoyé sur tout le réseau à l'adresse d'**un** destinataire,
- les **Réseaux en Mode Point à Point**,
  - une seule liaison entre deux équipements,
  - les nœuds permettent de choisir la route en fonction de l'adresse du destinataire,
  - quand deux nœuds non directement connectés entre eux veulent communiquer ils le font par l'intermédiaire des autres noeuds du réseau.

Classification par Topologie

> La **Topologie Physique** d'un réseau décrit l'organisation de ce dernier en termes de câblage. La **Topologie Logique** d'un réseau décrit comment les données circulent sur le réseau. En effet c'est le choix des concentrateurs ainsi que les connections des câbles qui déterminent la topologie logique.

La Topologie Physique

Il existe 6 topologies physiques de réseau :

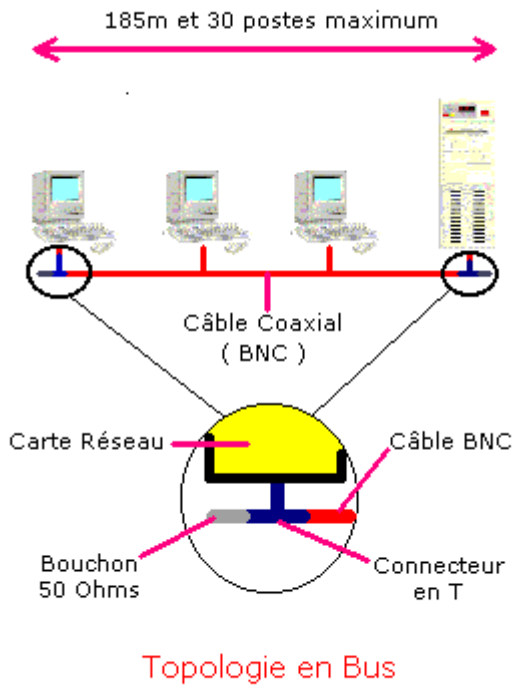
- La Topologie en Ligne,
- La Topologie en Bus,
- La Topologie en Etoile,
- La Topologie en Anneau,
- La Topologie en Arbre,
- La Topologie Maillée.

La Topologie en Ligne

Tous les nœuds sont connectés à un seul support. L'inconvénient de cette topologie est que dans le cas d'une défaillance d'une station, le réseau se trouve coupé en deux sous-réseaux.

La Topologie en Bus

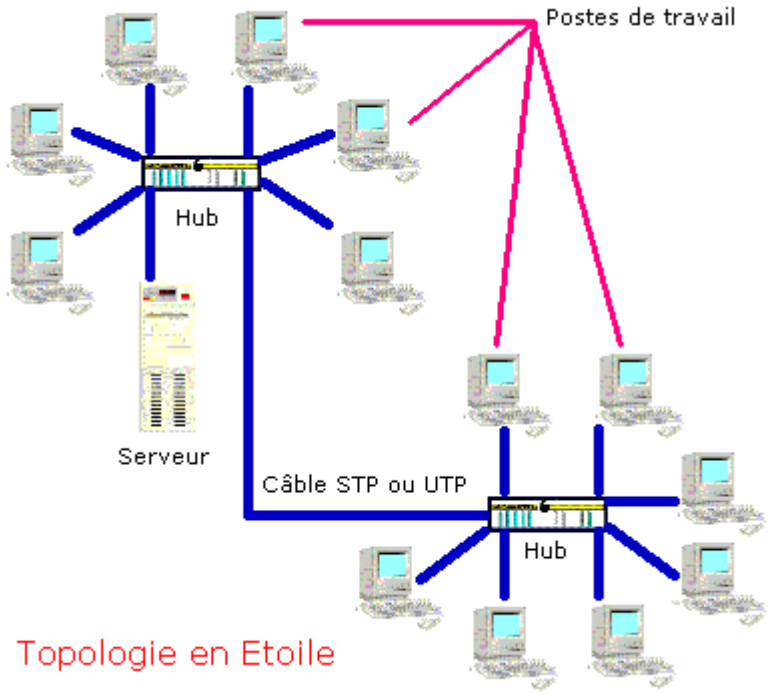
Tous les nœuds sont connectés à un seul support (un câble BNC en T) avec des bouchons à chaque extrémité. La longueur du bus est limitée à **185m**. Le nombre de stations de travail est limité à **30**. Les Stations sont reliées au Bus par des 'T'. Les bouchons sont des terminateurs qui sont des résistances de **50 Ohms**. Quand le support tombe en panne, le réseau ne fonctionne plus. Quand une station tombe en panne, elle ne perturbe pas le fonctionnement de l'ensemble du réseau. Les Stations étant reliés à un suel support, ce type de topologie necessite un **Protocole d'Accès** pour gérer le tour de parole des Stations afin d'éviter des conflits.



Topologie en Bus

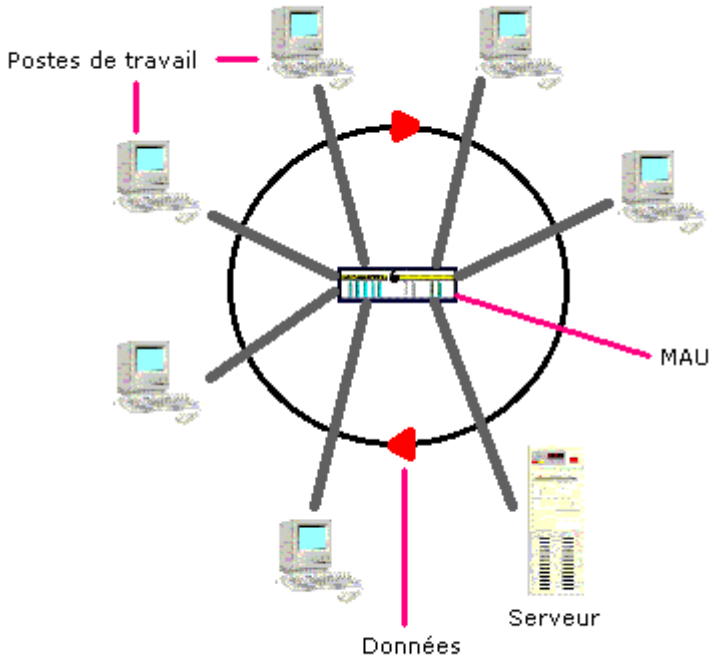
La Topologie en Étoile

Chaque nœud est connecté à un périphérique central appelé un **Hub** (**Concentrateur**) ou un **Switch** (**Commutateur**). Un Hub ou un Switch est prévu pour 4, 8, 16, 32 ... stations. En cas d'un réseau d'un plus grand nombre de stations, plusieurs Hubs ou Switches sont connectés ensemble. Quand une station tombe en panne, elle ne perturbe pas le fonctionnement de l'ensemble du réseau. Le point faible de cette topologie est l'équipement central.



La Topologie en Anneau

Chaque nœud est relié directement à ses deux voisins dans une topologie logique de cercle ininterrompu et une topologie physique en étoile car les stations sont reliées à un type de hub spécial, appelé un **Multistation Access Unit** (MAU).



Topologie en Anneau

Les stations sont reliées à la MAU par un câble 'IBM' munie d'une prise **AUI** du côté de la carte et une prise **Hermaphrodite** du coté de la MAU. Les données sont échangées dans un sens unidirectionnel. Une trame, appelée un **jeton**, circule en permanence. Si l'anneau est brisé, l'ensemble du réseau s'arrête. Pour cette raison, il est courant de voir deux anneaux contre-rotatifs.

La Topologie en Arbre

La Topologie en Arbre est utilisée dans un réseau hierarchique où le sommet, aussi appelé la **racine**, est connecté à plusieurs noeuds de niveau inférieur. Ces neouds peuvent à leur tour être connectés à d'autres noeuds inférieurs. L'ensemble forme une arborescence. Le point faible de cette topologie est sa racine. En cas de défaillance, le réseau est coupé en deux.

La Topologie Maillée

Cette Topologie est utilisée pour des grands réseaux de distribution tels Internet ou le WIFI. Chaque noeud à tous les autres via des liaisons point à point. Le nombre de liaisons devient très rapidement important en cas d'un grand nombre de noeuds. Par exemple dans le cas de 100 Stations (N), le nombre de liaisons est obtenu par la formule suivante :

```
N(N-1)/2 = 100(100-1)/2 = 4 950
```

La **Topologie Physique** la plus répandue est la **Topologie en Etoile**.

Classification par Etendue

La classification par étendue nous fournit 4 réseaux principaux :

Nom	Description	Traduction	Taille Approximative (M)
PAN	Personal Area Network	Réseau Personnel	1 -10
LAN	Local Area Network	Réseau Local Entreprise (RLE)	5 - 1 200
MAN	Métropolitain Area Network	Réseau Urbain	900 - 100 000
WAN	Wide Area Network	Réseau Long Distance (RLD)	50 000 et au delà

Cependant, d'autres classification existent :

CAN	Campus Area Network	Réseau de Campus
GAN	Global Area Network	Réseau Global
TAN	Tiny Area Network	Réseau Minuscule
FAN	Family Area Network	Réseau Familial
SAN	Storage Area Network	Réseau de Stockage

Etant donné que les WANs sont gérés par des opérateurs de télécommunications qui doivent demander une licence à l'état mais que les LANs ont été historiquement mis en oeuvre dans les entreprises, ces derniers sont en majorité issus du monde informatique.

Les Types de LAN

Il existe deux types de LAN :

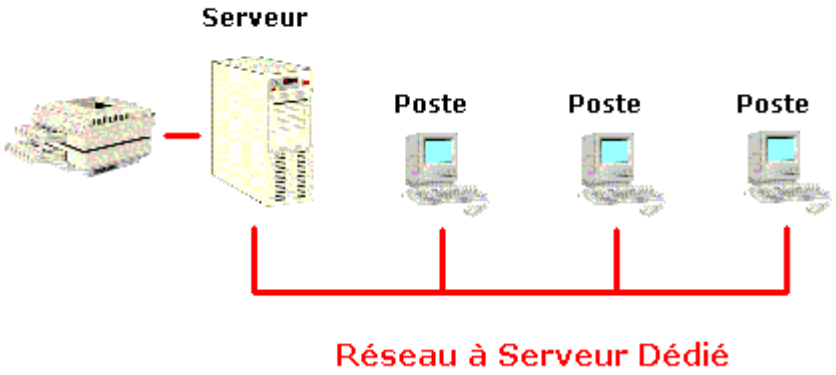
- le réseau à serveur dédié,
- le réseau poste à poste.

Réseau à Serveur Dédié

Le réseau à serveur dédié est caractérisé par le fait que toutes les ressources (imprimantes, applications, lecteurs etc.) sont gérées par le serveur. Les autres micro-ordinateurs ne jouent le rôle de client.

Des exemples des systèmes d'exploitation du réseau à serveur dédié sont :

- Windows NT Server,
- Windows 2000 Server,
- Windows 2003 Server,
- Windows 2008 Server,
- Linux,
- Unix.



Réseau Poste-à-Poste

Le réseau poste à poste est caractérisé par le fait que tous les ordinateurs peuvent jouer le rôle de client et de serveur :

- Windows 95,
- Windows 98,
- Windows NT Workstation.

Le Modèle Client/Serveur

Le modèle Client/Serveur est une des modalités des architectures informatiques distribuées. Dans ce modèle un serveur est tout **Logiciel** fournissant un **Service**.

Le serveur est aussi :

- passif, c'est-à-dire en attente permenante d'une demande, appelée une requête d'un client,
- capable de traiter plusieurs requêtes simultanément en utilisant le **multi-threading**,
- garant de l'intégrité globale.

Le client est, par contre **actif**, étant à l'origine des requêtes.

Il existe trois types de modèle client/serveur :

- **Plat** - tous les clients communiques avec un seul serveur,
- **Hiérarchique** - les clients n'ont de contact qu'avec les serveurs de plus haut niveau qu'eux,
- **Peer-to-Peer** - les équipements sont à la fois client **et** serveur en même temps.

Modèles de Communication

Les réseaux sont bâtis sur des technologies et des modèles. Le modèle **théorique** le plus important est le modèle **O**pen **S**ystem **Interconnection** créé par l'**I**nternational **Organization** for **S**tandardization tandis que le modèle pratique le plus important est le modèle **TCP/IP**.

Le modèle OSI

Le modèle OSI qui a été proposé par l'ISO est devenu le standard en termes de modèle pour décrire l'échange de données entre ordinateurs. Cette norme se repose sur sept couches, de la une - la Couche Physique, à la sept - la Couche d'Application, appelés des services. La communication entre les différentes couches est synchronisée entre le poste émetteur et le poste récepteur grâce à ce que l'on appelle un protocole.

Ce modèle repose sur trois termes :

- Les **Couches**,
- Les **Protocoles**,
- Les **Interfaces**.

Les Couches

Des sept couches :

- Les couches 1 à 3 sont les **Couches Basses** orientées **Transmission**,
- La couche 4 est la **Couche Charnière** entre les **Couches Basses** et les **Couches Hautes**,

- Les couches 5 à 7 sont les **Couches Hautes** orientées **Traitement**.

La couche du même niveau du système **A** parle avec son homologue du système **B**.

- **La Couche Physique** (Couche 1) est responsable :
  - du transfert de données binaires sur le câble physique ou virtuel
  - de la définition de tout aspect physique allant du connecteur jusqu'au câble en passant par la carte réseau, y compris l'organisation même du réseau
  - de la définition des tensions électriques sur le câble pour obtenir le 0 et le 1 binaires
- **La Couche de Liaison** (Couche 2) est responsable :
  - de la réception des données de la couche physique
  - de l'organisation des données en fragments, appelés des trames qui ont un format différent selon s'il s'agit d'un réseau basé sur la technologie Ethernet ou la technologie Token-Ring
  - de la préparation, émission et réception des trames
  - de la gestion de l'accès au réseau
  - de la communication nœud à nœud
  - de la gestion des erreurs
    - avant la transmission, le nœud émetteur calcule un code appelé un CRC et l'incorpore dans les données envoyées
    - le nœud récepteur recalcule un CRC en fonction du contenu de la trame reçue et le compare à celui incorporé avec l'envoi
    - en cas de deux CRC identique, le nœud récepteur envoie un accusé de réception au nœud émetteur
  - de la réception de l'accusé de réception
  - éventuellement de le ré-émission des données
  - En prenant ce modèle, l'IEEE (Institute of Electrical and Eletronics Engineers) l'a étendu avec le Modèle IEEE (802).
    - Dans ce modèle la Couche de Liaison est divisée en deux sous-couches importantes :
      - La **Sous-Couche LLC** (Logical Link Control) qui :
        - gère les accusés de réception
        - gère le flux de trames
      - La **Sous-Couche MAC** (Media Access Control) qui :
        - gère la méthode d'accès au réseau
        - le CSMA/CD dans un réseau basé sur la technologie Ethernet
        - l'accès au jeton dans un réseau basé sur la technologie Token-Ring
        - gère les erreurs
- **La Couche de Réseau** (Couche 3) est responsable de la gestion de la bonne distribution des différentes informations aux bonnes adresses en :

    - identifiant le chemin à emprunter d'un nœud donné à un autre
    - appliquant une conversion des adresses logiques (des noms) en adresses physiques
    - ajoutant des information adressage aux envois
    - détectant des paquets trop volumineux avant l'envoi et en les divisant en trames de données de tailles autorisées

- **La Couche de Transport** (Couche 4) est responsable de veiller à ce que les données soient envoyées correctement en :
    - constituant des paquets de données corrects
    - les envoyant dans le bon ordre
    - vérifiant que les données sont traités dans le même ordre que l'ordre d'émission
    - permettant à un processus sur un nœud de communiquer avec un autre nœud et d'échanger des messages avec lui

- **La Couche de Session** (Couche 5) est responsable :
    - de l'établissement, du maintien, et de la mise à fin de la communication entre deux noeuds distants, c'est-à-dire, de la session
    - de la conversation entre deux processus de vérification de la réception des messages envoyés en séquences, c'est-à-dire, le point de contrôle

- de la sécurité lors de l'ouverture de la session, c'est-à-dire, les droits d'utilisateurs etc.

- **La Couche de Présentation** (Couche 6) est responsable :
    - du formatage et de la mise en forme des données
    - des conversions de données telles le cryptage/décryptage

- **La Couche d'Application** (Couche 7) est responsable :
    - du dialogue homme/machine via des messages affichés
    - du partage des ressources
    - de la messagerie

Les Protocoles

Un **protocole** est un langage commun utilisé par dexu entités en communication pour pouvoir se comprendre. La nature du Protocole dépends directement de la nature de la communication. Cette bature dépend du **paradigme** de communication que l'application nécessite. Le paradigme est un modèle abstrait d'un problème ou d'une situation. Dans le paradigme de la diffusion, l'émetteur envoie dans informations au récepteur sans se soucier de ce que le récepteur va en faire. C'est la responsabilité du récepteur de comprendre et d'utiliser les informations.

Les Interfaces

Chaque couche rend des **services** à la couche immédiatement supérieure et utilise les services de la couche immédiatement inférieure. L'ensemble des services s'appelle une **Interface**. Les services sont composés de **S**ervice **D**ata **U**nits et sont disponibles par un **S**service **A**ccess **P**oint.

Protocol Data Units

L'**Unité de Données** ou *Protocol Data Unit* pour chaque couche comporte un nom spécifique :

- **Application Protocol Data Units** pour la couche **Application**,
- **Présentation Protocol Data Units** pour la couche **Présentation**,
- **Session Protocol Data Units** pour la couche **Session**,
- **Transport Protocol Data Units** pour la couche **Transport**.

Or, pour les **Couches Basses** on parle de :

- **Paquets** pour la couche **Réseau**,
- **Trames** pour la couche **Liaison**,
- **Bits** pouyr la couche **Physique**.

Encapsulation et Désencapsulation

Lorque les données sont communiqueés par le système A au système B, celles-ci commencent au niveau de la couche d'Application. Le couche d'Application ajoute une en-tête à l'unité de données qui contient des **informations de contrôle du protocole**. Au passage de chaque couche, celle-ci ajoute sa propre en-tête. De cette façon, lors de sa descente vers la couche physique, les données et l'entête de la couche supérieure sont encapulsulées :

Couche Système A	**Encapsulation**
Application	Application Header (AH) + Unité de Données (UD)
Présentation	Présentation Header (PH) + AH + UD
Session	Session Header (SH) + PH + AH + UD

Couche Système A	Encapsulation
Transport	Transport Header (TH) + SH + PH + AH + UD
Réseau	Network Header (NH) + TH + SH + PH + AH + UD
Liaison	Liaison Header (DH) + NH + TH + SH + PH + AH + UD

Lors de son voyage de la couche Physique vers la couche Application dans le système B, les en-têtes sont supprimées par chaque couche correspondante. On parle alors de **désencapsulation** :

Couche Système B	Encapsulation
Liaison	Liaison Header (DH) + NH + TH + SH + PH + AH + UD
Réseau	Network Header (NH) + TH + SH + PH + AH + UD
Transport	Transport Header (TH) + SH + PH + AH + UD
Session	Session Header (SH) + PH + AH + UD
Présentation	Présentation Header (PH) + AH + UD
Application	Application Header (AH) + Unité de Données (UD)

Spécification NDIS et le Modèle ODI

<note tip> **Cliquez ici pour ouvrir le schéma Simplifié du Modèle OSI incluant la spécification NDIS** </note>

La spécification NDIS (Network Driver Interface Specification) a été introduite conjointement par les sociétés Microsoft et 3Com. Cette spécification ainsi que son homologue, le modèle ODI (Open Datalink Interface) introduit conjointement par les sociétés Novell et Apple à la même époque, définit des standards pour les pilotes de cartes réseau afin qu'ils puissent être indépendants des protocoles utilisées et les systèmes d'exploitation sur les machines. Des deux 'standards', la spécification NDIS est le plus répandu, intervenant a niveau de la sous-couche MAC et l a couche de liaison. Elle spécifie :

- l'interface pilote-matériel
- l'interface pilote-protocole
- l'interface pilote - système d'exploitation

Le modèle TCP/IP

<note tip> **Cliquez ici pour voir le modèle OSI incluant la suite des protocoles et services TCP/IP** </note>

La suite des protocoles TCP/IP (Transmission Control Protocol / Internet Protocol) est issu de la DOD (Dept. Américain de la Défense) et le travail de l'ARPA (Advanced Research Project Agency).

- La suite des protocoles TCP/IP
  - a été introduite en 1974
  - a été utilisée dans l'ARPAnet en 1975
  - permet la communication entre des réseaux à base de systèmes d'exploitation, architectures et technologies différents
  - est très proche du modèle OSI en termes d'architecture et se place au niveau de la couche d'Application jusqu'à la couche Réseau.
  - est, en réalité, une suite de protocoles et de services :
    - **IP** (Internet Protocol)
      - le protocole IP s'intègre dans la couche Réseau du modèle OSI en assurant la communication entre les systèmes. Bien qu'il puisse découper des messages en fragments ou datagrammes et les reconstituer dans le bon ordre à l'arrivée, il ne garantit pas la réception.
    - **ICMP** (Internet Control Message Protocol)
      - le protocole ICMP produit des messages de contrôle aidant à synchroniser le réseau. Un exemple de ceci est la commande ping.
    - **TCP** (Transmission Control Protocol)
      - le protocole TCP se trouve au niveau de la couche de Transport du modèle OSI et s'occupe de la transmission des données entre noeuds.
    - **UDP** (User Datagram Protocol)
      - le protocole UDP n'est pas orienté connexion. Il est utilisé pour la transmission rapide de messages entre nœuds sans garantir leur acheminement.
    - **Telnet**
      - le protocole Telnet est utilisé pour établir une connexion de terminal à distance. Il se trouve dans la couche d'Application du modèle OSI.
    - **Ftp** (File Transfer Protocol)
      - le protocole ftp est utilisé pour le transfert de fichiers. Il se trouve dans la couche d'Application du modèle OSI.
    - **SMTP** (Simple Message Transfer Protocol)
      - le service SMTP est utilisé pour le transfert de courrier électronique. Il se trouve dans la couche d'Application du modèle OSI.
    - **DNS** (Domain Name Service)
      - le service DNS est utilisé pour le résolution de noms en adresses IP. Il se trouve dans la couche d'Application du modèle OSI.
    - **SNMP** (Simple Network Management Protocol)

        - le protocole SNMP est composé d'un agent et un gestionnaire. L'agent SNMP collecte des informations sur les périphériques, les configurations et les performances tandis que le gestionnaire SNMP reçois ses informations et réagit en conséquence.
    - **NFS** (Network File System)
        - le NFS a été mis au point par Sun Microsystems
        - le NFS génère un lien virtuel entre les lecteurs et les disques durs permettant de monter dans un disque virtuel local un disque distant
    - et aussi POP3, NNTP, IMAP etc ...

<note tip> **Cliquez ici pour voir les modèles TCP/IP et OSI** </note>

Le modèle TCP/IP est composé de 4 couches :

- La couche d'Accès Réseau
    - Cette couche spécifie la forme sous laquelle les données doivent être acheminées, quelque soit le type de réseau utilisé.
- La couche Internet
    - Cette couche est chargée de fournir le paquet de données.
- La couche de Transport
    - Cette couche assure l'acheminement des données et se charge des mécanismes permettant de connaître l'état de la transmission.
- La couche d'Application
    - Cette couche englobe les applications standards de réseau telles ftp, telnet, ssh, etc..

Les noms des Unités de Données sont différents selon le protocole utilisé et la couche du modèle TCP/IP :

Couche	**TCP**	**UDP**
Application	Stream	Message
Transport	Segment	Packet
Internet	Datagram	Datagram
Réseau	Frame	Frame

Les Raccordements

Les Modes de Transmission

On peut distinguer 3 modes de transmission :

- La **Liaison Simplex**,
  - Les données ne circulent que dans un **seul** sens de l'émetteur ver le récepteur,
  - La liaison nécessite deux canaux de transmissions,
- La **Liaison Half-Duplex** aussi appelée la **Liaison à l'Alternat** ou encore la **Liaison Semi-Duplex**,
  - Les données circulent dans un sens ou l'autre mais jamais dans les deux sens en même temps. Chaque extrémité émet donc à son tour,
  - La liaison permet d'avoir une liaison bi-directionnelle qui utilise la totalité de la banse passante,
- La **Liaison Full-Duplex** dans les deux sens en **même** temps. Chaque extrémité peut émettre et recevoir simultanément,
  - La liaison est caractérisée par une bande passante divisée par deux pour chaque sens des émissions.

Les Câbles

Le Câble Coaxial

En partant de l'extérieur, le câble coaxial est composé :

- d'une **Gaine** en caoutchouc, PVC ou Téflon pour protéger le câble,
- d'un **Blindage** en métal pour diminuer le bruit du aux parasites,
- d'un **Isolant** (diélectrique) pour éviter le contact entre le blindage et l'âme et ainsi éviter des courts-circuits,
- d'un **Âme** en cuivre ou torsadés pour transporter les données.

Avantages :

- **Peux coûteux**,
- Facilement **manipulable**,
- Peut être utilisé pour de **longues distances**,
- A un débit de 10 Mbit/s dans un LAN et 100 Mbit/s dans un WAN.

Inconvénients :

- Fragile,
- Instable,

- Vulnérable aux interférences,
- Half-Duplex.

Le Câble Paire Torsadée

Ce câble existe sous deux formes selon son utilisation :

- **Monobrin** pour du câblage **horizontal** (**Capillaire**),
  - chaque fil est composé d'un seul conducteur en cuivre,
  - la distance ne doit pas dépassée 90m.
- **Multibrin** pour des **cordons de brassage** :
  - chaque fil est composé de plusieurs brins en cuivre,
  - câble souple.

Avantages :

- Un débit de 10 Mbit/s à 10 GBit/s,
- A une bande passante plus large,
- Pas d'interruption par coupure du câble,
- Permet le **câblage universel** (téléphonie, fax, données ...),
- Full-Duplex.

Inconvénients :

- Nombre de câbles > câble coaxial,
- Plus cher,
- Plus encombrant dans les gaines techniques.

Catagories de Blindage

Il existe trois catagories de blindage :

- **Twisted** ou Torsadé,
- **Foiled** ou Entouré,
- **Shielded** ou Avec Ecran.

De ce fait, il existe 5 catagories de câbles Paire Torsadée :

Nom anglais ^ Appelation Ancienne ^ Nouvelle Appelation ^

Unshielded Twisted Pair	UTP	U/UTP
Foiled Twisted Pair	FTP	F/UTP
Shield Twisted Pair	STP	S/UTP
Shield Foiled Twisted Pair	SFTP	SF/UTP
Shield Shield Twisted Pair	S/STP	SS/STP3

Ces catégories donnent lieu à des **Classes** :

Classe	**Débit**	**Nombre de Paires Torsadées**	**Connecteur**	**Commentaires**
3	10 Mbit/s	4	RJ11	
4	16 Mbit/s	4	S/O	Non-utilisée de nos jours
5	100 Mbit/s	4	RJ45	Obsolète
5e/D	1 Gbit/s sur 100m	4	RJ45	S/O
6/E	2.5 Gbit/s sur 100m ou 10 Gbit/s sur 25m à 55m	4	Idéal pour PoE	
7/F	10 Gbit/s sur 100m	4	GG45 ou Tera	Paires individuellement et collectivement blindées. Problème de compatibilité avec les classes precédentes due au connecteur.

La Prise RJ45

Une prise RJ45 comporte 8 broches. Un câble peut être **droit** quand la broche 1 d'une extremité est connectée à la broche 1 de la prise RJ45 à l'autre extrémité, la broche 2 d'une extremité est connectée à la broche 2 de la prise RJ45 à l'autre extrémité et ainsi de suite ou bien **croisé** quand le brochage est inversé.

Les câbles croisés sont utilisés lors du branchement de deux équipements identiques (PC à PC, Hub à Hub, Routeur à Routeur).

Channel Link et Basic Link

Le **Channel Link** ou **Canal** est l'ensemble du **Basic Link** ou **Lien** de base et les cordons de brassage et de raccordement des équipements qui sont limités en distance à 10m.

Le **Basic Link** est le lien entre la prise RJ45 murale et la baie de brassage. Il est limité à 90m en classe 5D.

La Fibre Optique

La **Fibre Optique** est un fil de **Silice** permettant le transfert de la lumière. De ce fait elle est caractérisée par :

- des meilleures performances que le cuivre,
- de plus de communications simultanément,
- de la capacité de relier de plus grandes distances,
- une insensibilité aux perturbations,
- une résistance à la corrosion.

Qui plus est, elle ne produit aucune perturbation.

Elle est composée :

- d'un coeur de 10, de 50/125 ou de 62.50 micron,
- d'une gaine de 125 micron,
- d'une protection de 230 micron.

Il existe deux types de fibres, la **Fibre Monomode** et la **Fibre Multimodes**.

La Fibre Monomode :

- a un coeur de 8 à 10 Microns,
- est divisée en sous-catégories de distance,

    - 10 Km,
    - 15 Km,
    - 20 Km,
    - 50 Km,
    - 80 Km,
    - 100 Km.

La Fibre Multimode :

- a un coeur de 62,50 micron ou de 50/125 micron avec une gaine orange,
- permet plusieurs trajets lumineux appelés **modes** en même temps en Full Duplex,
- est utilisée pour de bas débits ou de courtes distances,
    - 2 Km pour 100 Mbit/s,
    - 500 m pour 1 Gbit/s.

Les Réseaux sans Fils

Les réseaux sans fils sans basés sur une liaison qui utilise des ondes radio-électriques (radio et infra-rouges).

Il existe des technologies différentes en fonction de la fréquence utilisée et de la portée des transmissions :

- Réseaux Personnels sans Fils - Bluetooth, HomeRF,
- Réseaux Locaux sans Fils - LiFI, WiFI,
- Réseaux Métropolitains sans Fil - wImax,
- Réseaux Etendus sans Fils - GSM, GPRS, UMTS.

Les principales ondes utilisées pour la transmission des données sont :

- Ondes GSM - Ondes Hertziennes repeosant sur des micro-ondes à basse fréquence avec une portée d'une dizaine de kilomètres,
- Ondes Wi-Fi - Ondes Hertziennes reposant sur des micro-ondes à haute fréquence avec une portée de 20 à 50 mètres,
- Ondes Satellitaires - Ondes Hertziennes longues portées.

Le Courant Porteur en Ligne

Le CPL utilise le réseau électrique domestique, le réseau moyenne et basse tension pour transmettre des informations numériques.

Le CPL superpose un signal à plus haute fréquence au signal électrique.

Seuls donc, les fils conducteurs transportent les signaux CPL.

Le coupleur intégré en entrée des boîtiers CPL élimine les composants basses fréquences pour isoler le signal CPL.

Le CPL utilise la phase électrique et le neutre. De ce fait, une installation triphasée fournit 3 réseaux CPL différents.

Le signal CPL ne s'arrête pas necéssairement aux limites de l'installation électrique. En effet en cas de compteurs non-numériques le signal les traversent.

Les normes CPL sont :

Norme	Débit Théorique	Débit Pratique	Temps pour copier 1 Go
Homeplug 1.01	14 Mbps	5.4 Mbps	25m 20s
Homeplug 1.1	85 Mbps	12 Mbps	11m 20s
PréUPA 200	200 Mbps	30 Mbps	4m 30s

Technologies

Il existe plusieurs technologies de réseau :

- Ethernet,
- Token-Ring,
- ARCnet,
- etc..

Nous détaillerons ici les deux technologies les plus répandues, à savoir Ethernet et Token-Ring.

Ethernet

La technologie Ethernet se repose sur :

- une topologie logique de bus,
- une topologie physique de bus ou étoile.

L'accès au bus utilise le **CSMA/CD**, Carrier Sense Multiple Access / Collision Detection (Accès Multiple à Détection de Porteuse / Détection de Collisions).

Il faut noter que :

- les données sont transmises à chaque nœud - c'est la méthode d'**accès multiple**,
- chaque nœud qui veut émettre écoute le réseau - c'est la **détection de porteuse**,
- quand le réseau est silencieux une trame est émise dans laquelle se trouvent les données ainsi que l'adresse du destinataire,
- le système est dit donc **aléatoire** ou **non-déterministe**,
- quand deux nœuds émettent en même temps, il y a **collision de données**,
- les deux nœuds vont donc cesser d'émettre, se mettant en attente jusqu'à ce qu'ils commencent à émettre de nouveau.

Token-Ring

La technologie Token-Ring se repose sur :

- une topologie logique en anneau,
- une topologie physique en étoile.

Token-Ring se traduit par **Anneau à Jeton**. Il n'est pas aussi répandu que l'Ethernet pour des raisons de coûts. En effet le rajout d'un nœud en Token-Ring peut coûter jusqu'à **4 fois plus cher qu'en Ethernet**.

Il faut noter que :

- les données sont transmises dans le réseau par un système appelé **méthode de passage de jeton**,
- le jeton est une **trame numérique vide** de données qui tourne en permanence dans l'anneau,
- quand un nœud souhaite émettre, il saisit le jeton, y dépose des données avec l'adresse du destinataire et ensuite laisse poursuivre son chemin

jusqu'à sa destination,
- pendant son voyage, aucun autre nœud ne peut émettre,
- une fois arrivé à sa destination, le jeton dépose ses données et retourne à l'émetteur pour confirmer la livraison,
- ce système est appelé **déterministe**.

L'intérêt de la technologie Token-Ring se trouve dans le fait :

- qu'il **évite des collisions**,
- qu'il est **possible de déterminer avec exactitude le temps que prenne l'acheminement des données**.

La technologie Token-Ring est donc idéale, voire obligatoire, dans des installations où chaque nœud doit disposer d'une opportunité à intervalle fixe d'émettre des données.

Périphériques Réseaux Spéciaux

En plus du câblage, les périphériques de réseau spéciaux sont des éléments primordiaux tant au niveau de la topologie physique que la topologie logique.

Les périphériques de réseau spéciaux sont :

- les Concentrateurs ou *Hubs*,
- les Répéteurs ou *Repeaters*,
- les Ponts ou *Bridges*,
- les Commutateurs ou *Switches*,
- les Routeurs ou *Routers*,
- les Passerelles ou *Gateways*.

L'objectif ici est de vous permettre de comprendre le rôle de chaque périphérique.

Les Concentrateurs

Les Concentrateurs permettent une connectivité entre les nœuds en topologie en étoile. Selon leur configuration, la topologie logique peut être en

étoile, en bus ou en anneau. Il existe de multiples types de Concentrateurs allant du plus simple au Concentrateur intelligent.

- **Le Concentrateur Simple**
  - est une boîte de raccordement centrale,
  - joue le rôle de récepteur et du réémetteur des signaux sans accélération ni gestion de ceux-ci,
  - est un périphérique utilisé pour des groupes de travail.
- **Le Concentrateur Évolué**
  - est un Concentrateur simple qui offre en plus l'amplification des signaux, la gestion du type de topologie logique grâce à des capacités d'être configurés à l'aide d'un logiciel ainsi que l'homogénéisation du réseau en offrant des ports pour un câblage différent. Par exemple, 8 ports en paire torsadée non-blindée et un port BNC.
- **Le Concentrateur Intelligent**
  - est un Concentrateur évolué qui offre en plus la détection automatique des pannes, la connectique avec un Pont ou un Routeur ainsi que le diagnostic et la génération de rapports.

Les Répéteurs

Un Répéteur est un périphérique réseau simple. Il est utilisé pour amplifier le signal quand :

- la longueur du câble dépasse la limite autorisée,
- le câble passe par une zone ou les interférences sont importantes.

Éventuellement, et uniquement dans le cas où le Répéteur serait muni d'une telle fonction, celui-ci peut être utiliser pour connecter deux réseaux ayant un câblage différent.

Les Ponts

Un Pont est **Répéteur intelligent**. Outre sa capacité d'amplifier les signaux, le Pont analyse le trafic qui passe par lui et met à jour une liste d'adresses des cartes réseau, appelée **une table de routage**, n'autorisant que les transmissions destinées à d'autres segments du réseau.

Les **diffusions** sont néanmoins autorisées.

Comme un Pont doit être intelligent, on utilise souvent un micro-ordinateur comme Pont. Forcément équipé de 2 cartes réseau, le Pont peut également jouer le rôle de serveur de fichiers.

Le Pont sert donc à isoler des segments du réseau pour des raisons de :

- **sécurité** afin d'éviter à ce que des données sensibles soient propagées sur tout le réseau,
- **performance** afin qu'une partie du réseau trop chargée ralentisse le réseau entier,
- **fiabilité** afin par exemple qu'une carte en panne ne gène pas le reste du réseau avec une diffusion.

Il existe trois types de configuration de Ponts

Le Pont de Base

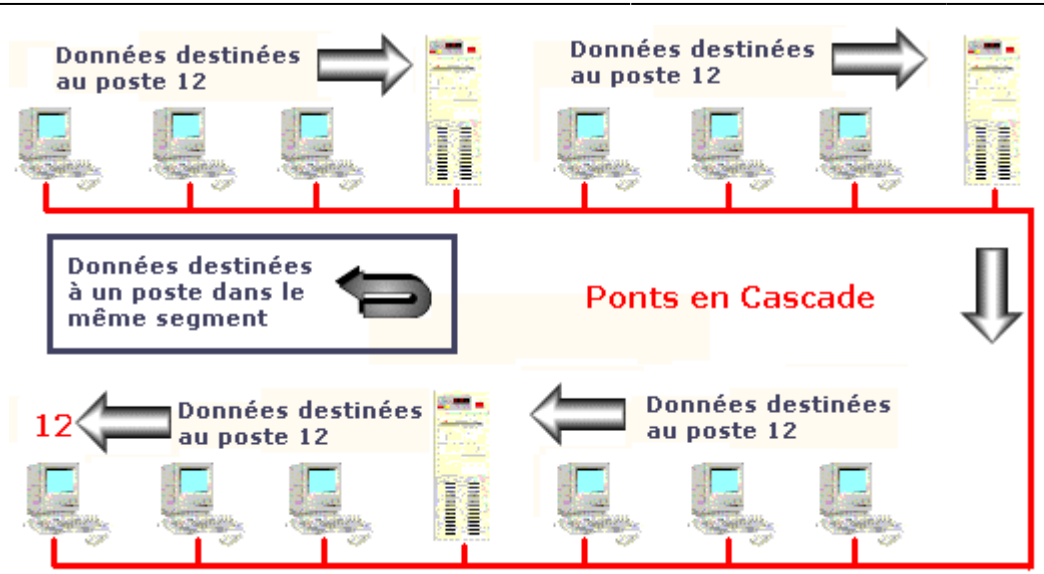
Le Pont de Base est utilisé très rarement pour isoler deux segments.

Pont de Base

;#; ;#;

Le Pont en Cascade

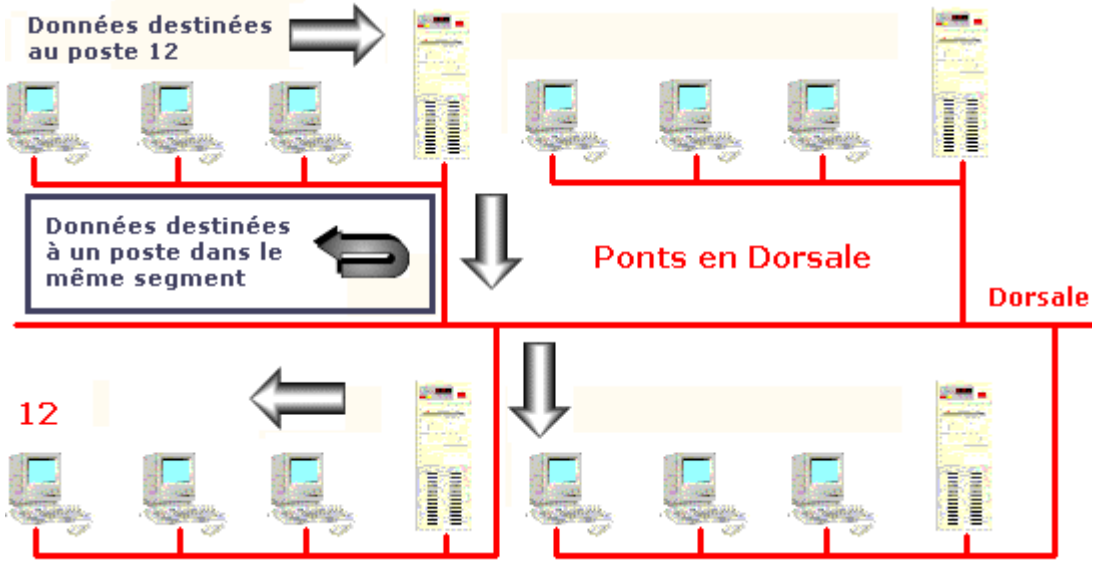
Le Pont en Cascade est à éviter car les données en provenance d'un segment doivent passer par plusieurs Ponts. Ceci a pour conséquence de ralentir la transmission des données, voire même de créer un trafic superflu en cas de rémission par le nœud



;#; ;#;

Le Pont en Dorsale

Le Pont en Dorsale coûte plus chère que la configuration précédente car il faut un nombre de Ponts équivalent au nombre de segments + 1. Par contre elle réduit les problèmes précédemment cités puisque les données ne transitent que par deux Ponts.



;#; ;#;

Les Commutateurs

Un Commutateur peut être considéré comme un Concentrateur intelligent et un Pont. Ils sont gérés souvent par des logiciels. La topologie physique d'un réseau commuté est en étoile. Par contre la topologie logique est spéciale, elle s'appelle une topologie commutée.

Lors de la communication de données entre deux nœuds, le Commutateur ouvre une connexion temporaire virtuelle en fermant les autres ports. De cette façon la bande passante totale est disponible pour cette transmission et les risques de collision sont minimisés.

Certains Commutateurs haut de gamme sont équipés d'un système anti-catastrophe qui leur permet d'isoler une partie d'un réseau en panne afin que les autres parties puissent continuer à fonctionner sans problème.

Les Routeurs

Un Routeur est un Pont sophistiqué capable :

- d'assurer l'interconnexion entre des segments,
- de filtrer le trafic,
- d'isoler une partie du réseau,
- d' explorer les informations d'adressage pour trouver le chemin le plus approprié et le plus rentable pour la transmission des données.

Les Routeurs utilisent une table de routage pour stocker les informations sur :

- les adresses du réseau,
- les solutions de connexion vers d'autres réseaux,
- l'efficacité des différentes routes.

Il existe deux types de Routeur :

- le **Routeur Statique**
  - la table de routage est éditer manuellement,
  - les routes empruntées pour la transmission des données sont toujours les mêmes,
  - il n'y a pas de recherche d'efficacité.
- le **Routeur Dynamique**
  - découvre automatiquement les routes à emprunter dans un réseau.

Les Passerelles

Ce périphérique, souvent un logiciel, sert à faire une conversion de données :

- entre deux technologies différentes (Ethernet - Token-Ring),
- entre deux protocoles différents,
- entre des formats de données différents.

Annexe #2 - Comprendre TCP Version 4

En-tête TCP

L'en-tête TCP est codée sur 4 octets soit 32 bits :

<table>
<tr><th>1er octet</th><th>2ème octet</th><th>3ème octet</th><th>4 ème octet</th></tr>
<tr><td colspan="2">Port source</td><td colspan="2">Port destination</td></tr>
<tr><td colspan="4">Numéro de séquence</td></tr>
<tr><td colspan="4">Numéro d'acquittement</td></tr>
<tr><td>Offset</td><td>Flags</td><td colspan="2">Fenêtre</td></tr>
<tr><td colspan="2">Checksum</td><td colspan="2">Pointeur Urgent</td></tr>
<tr><td colspan="3">Options</td><td>Padding</td></tr>
<tr><td colspan="4">Données</td></tr>
</table>

Vous noterez que les numéros de ports sont codés sur 16 bits. Cette information nous permet de calculer le nombres de ports maximum en IPv4, soit 2^{16} ports ou 65 535.

L'**Offset** contient la taille de l'en-tête.

Les **Flags** sont :

- URG - Si la valeur est 1 le pointeur urgent est utilisé. Le numéro de séquence et le pointeur urgent indique un octet spécifique.
- ACK - Si la valeur est 1, le paquet est un accusé de réception
- PSH - Si la valeur est 1, les données sont immédiatement présentées à l'application
- RST - Si la valeur est 1, la communication comporte un problème et la connexion est réinitialisée
- SYN - Si la valeur est 1, le paquet est un paquet de synchronisation
- FIN - Si la valeur est 1, le paquet indique la fin de la connexion

La **Fenêtre** est codée sur 16 bits. La Fenêtre est une donnée liée au fonctionnement d'expédition de données appelé le **sliding window** ou la **fenêtre glissante**. Puisque il serait impossible, pour des raisons de performance, d'attendre l'accusé de réception de chaque paquet envoyé, l'expéditeur envoie des paquets par groupe. La taille de cette groupe s'appelle la Fenêtre. Dans le cas d'un problème de réception d'une partie de la Fenêtre, toute la Fenêtre est ré-expédiée.

Le **Checksum** est une façon de calculer si le paquet est complet.

Le **Padding** est un champ pouvant être rempli de valeurs nulles de façon à ce que la taille de l'en-tête soit un multiple de 32

En-tête UDP

L'en-tête UDP est codée sur 4 octets soit 32 bits :

1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Port source		Port destination	
longueur		Checksum	
Données			

L'en-tête UDP a une longueur de 8 octets.

Fragmentation et Ré-encapsulation

La taille limite d'un paquet TCP, l'en-tête comprise, ne peut pas dépasser **65 535 octets**. Cependant chaque réseau est qualifié par son MTU (Maximum Tranfer Unit). Cette valeur est la taille maximum d'un paquet autorisée. L'unité est en **octets**. Pour un réseau Ethernet sa valeur est de 1 500. Quand un paquet doit être expédié sur un réseau ayant un MTU inférieur à sa propre taille, le paquet doit être **fractionné**. A la sortie du réseau, le paquet est reconstitué. Cette reconstitution s'appelle **ré-encapsulation**.

Adressage

L'adressage IP requière que chaque périphérique sur le réseau possède une adresse IP unique de 4 octets, soit 32 bits au format XXX.XXX.XXX.XXX De cette façon le nombre total d'adresses est de 2^{32} = 4.3 Milliards.

Les adresses IP sont divisées en 5 classes, de A à E. Les 4 octets des classes A à C sont divisés en deux, une partie qui s'appelle le **Net ID** qui identifie le réseau et une partie qui s'appelle le **Host ID** qui identifie le hôte :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
A	Net ID	Host ID		

	1er octet	2ème octet	3ème octet	4 ème octet
B	Net ID		Host ID	
C	Net ID			Host ID
D	Multicast			
E	Réservé			

L'attribution d'une classe dépend du nombre de hôtes à connecter. Chaque classe est identifié par un **Class ID** composé de 1 à 3 bits :

Classe	Bits ID Classe	Valeur ID Classe	Bits ID Réseau	Nb. de Réseaux	Bits ID hôtes	Nb. d'adresses	Octet de Départ
A	1	0	7	2^7=128	24	2^{24}=16 777 216	1 - 126
B	2	10	14	2^{14}=16 834	16	2^{16}=65 535	128 - 191
C	3	110	21	2^{21}=2 097 152	8	2^8=256	192 - 223

Dans chaque classe, certaines adresses sont réservées pour un usage privé :

Classe	IP de Départ	IP de Fin
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

Il existe des adresses particulières ne pouvant pas être utilisées pour identifier un hôte :

Adresse Particulière	Description
169.254.0.0 à 169.254.255.255	Automatic Private IP Addressing de Microsoft
Hôte du réseau courant	Tous les bits du Net ID sont à 0
Adresse de réseau	Tous les bits du Host ID sont à 0
Adresse de diffusion	Tous les bits du Host ID sont à 1

L'adresse de réseau identifie le **segment** du réseau entier tandis que l'adresse de diffusion identifie tous les hôtes sur le segment de réseau.

Afin de mieux comprendre l'adresse de réseau et l'adresse de diffusion, prenons le cas de l'adresse 192.168.10.1 en classe C :

	1er octet	2ème octet	3ème octet	4 ème octet

	1er octet	2ème octet	3ème octet	4 ème octet

	Net ID			Host ID
Adresse IP	192	168	10	1
Binaire	11000000	10101000	000001010	00000001
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	000001010	**00000000**
Adresse réseau	192	168	10	0
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	000001010	**11111111**
Adresse de diffusion	192	168	10	255

Masques de sous-réseaux

Tout comme l'adresse IP, le masque de sous-réseau compte 4 octets ou 32 bits. Les masques de sous-réseaux permettent d'identifer le Net ID et le Host ID :

Classe	Masque	Notation CIDR
A	255.0.0.0	/8
B	255.255.0.0	/16
C	255.255.255.0	/24

Le terme **CIDR** veut dire **Classless InterDomain Routing**. Le terme Notation CIDR correspond au nombre de bits d'une valeur de 1 dans le masque de sous-réseau.

Quand un hôte souhaite émettre il procède d'abord à l'identification de sa propre adresse réseau par un calcul AND (ET) appliqué à sa propre adresse et son masque de sous-réseau qui stipule :

- 1 x 1 = 1
- 0 x 1 = 0
- 1 x 0 = 0
- 0 x 0 = 0

Prenons le cas de l'adresse IP 192.168.10.1 ayant un masque de 255.255.255.0 :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Adresse IP	192	168	10	1
Binaire	11000000	10101000	00001010	00000001
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Cet hôte essaie de communiquer avec un hôte ayant une adresse IP de 192.168.10.10. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Adresse IP	192	168	10	10
Binaire	11000000	10101000	00001010	00001010
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Puisque l'adresse réseau est identique dans les deux cas, l'hôte émetteur présume que l'hôte de destination se trouve sur son réseau et envoie les paquets directement sur le réseau sans s'adresser à sa passerelle par défaut.

L'hôte émetteur essaie maintenant de communiquer avec avec un hôte ayant une adresse IP de 192.168.2.1. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Adresse IP	192	168	2	1
Binaire	11000000	10101000	00000010	00000001
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00000010	00000000

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse réseau	192	168	2	0

Dans ce cas, l'hôte émetteur constate que le réseau de destination 192.168.2.0 n'est pas identique à son propre réseau 192.168.10.0. Il adresse donc les paquets à la passerelle par défaut.

VLSM

Puisque le stock de réseaux disponibles sous IPv4 est presque épuisé, une solution a du être trouvée pour créer des sous-réseaux en attendant l'introduction de l'IPv6. Cette solution s'appelle le VLSM ou Variable Length Subnet Masks. Le VLSM exprime les masques de sous-réseaux au format CIDR.

Son principe est simple. Afin de créer des réseaux différents à partir d'une adresse réseau d'une classe donnée, il convient de réduire le nombre d'hôtes. De cette façon les bits 'libérés' du Host ID peuvent être utilisés pour identifier les sous-réseaux.

Pour illustrer ceci, prenons l'exemple d'un réseau 192.168.1.0. Sur ce réseau, nous pouvons mettre 2^8-2 soit 254 hôtes entre 192.168.1.1 au 192.168.1.254.

Supposons que nous souhaiterions diviser notre réseau en 2 sous-réseaux. Pour coder 2 sous-réseaux, il faut que l'on libère 2 bits du Host ID. Les deux bits libérés auront les valeurs binaires suivantes :

- 00
- 01
- 10
- 11

Les valeurs binaires du quatrième octet de nos adresses de sous-réseaux seront donc :

- 192.168.1.00XXXXXX
- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX
- 192.168.1.11XXXXXX

où les XXXXXX représentent les bits que nous réservons pour décrire les hôtes dans chacun des sous-réseaux.

Nous ne pouvons pas utiliser les deux sous-réseaux suivants :

- 192.168.1.00XXXXXX
- 192.168.1.11XXXXXX

car ceux-ci correspondent aux débuts de l'adresse réseau 192.168.1.0 et de l'adresse de diffusion 192.168.1.255.

Nous pouvons utiliser les deux sous-réseaux suivants :

- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX

Pour le premier sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #1	192	168	1	01XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	01**000000**
Adresse réseau	192	168	1	**64**
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	01**111111**
Adresse de diffusion	192	168	1	**127**

- L'adresse CIDR du réseau est donc 192.168.1.64/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6-2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.65 à 192.168.1.126

Pour le deuxième sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #2	192	168	1	10XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	10**000000**
Adresse réseau	192	168	1	**128**

Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	10**111111**
Adresse de diffusion	192	168	1	**191**

- L'adresse CIDR du réseau est donc 192.168.1.128/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6-2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.129 à 192.168.1.190

La valeur qui sépare les sous-réseaux est 64. Cette valeur comporte le nom **incrément**.

Ports et sockets

Afin que les données arrivent aux applications que les attendent, TCP utilise des numéros de ports sur la couche transport. Le numéros de ports sont divisés en trois groupes :

- **Well Known Ports**
  - De 1 à 1023
- **Registered Ports**
  - De 1024 à 49151
- **Dynamic** et/ou **Private Ports**
  - De 49152 à 65535

Le couple **numéro IP:numéro de port** s'appelle un **socket**.

/etc/services

Les ports les plus utilisés sont détaillés dans le fichier **/etc/services** :

```
trainee@debian8:~$ su -
```

```
Password:
root@debian8:~# more /etc/services
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.
#
# Updated from http://www.iana.org/assignments/port-numbers and other
# sources like http://www.freebsd.org/cgi/cvsweb.cgi/src/etc/services .
# New ports will be added on request if they have been officially assigned
# by IANA and used in the real-world or are needed by a debian package.
# If you need a huge list of used numbers please install the nmap package.

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
--More--(6%)
```

Notez que les ports sont listés par deux :

- le port TCP
- le port UDP

La liste la plus complète peut être consultée à l'adresse suivante **https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml**.

Pour connaitre la liste des sockets ouverts sur l'ordinateur, saisissez la commande suivante :

```
root@debian8:~# netstat -an | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:42370         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:15023         0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:41012           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:7127          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:33220         127.0.0.1:50656         ESTABLISHED
tcp        0      0 10.0.2.15:22            10.0.2.2:46432          ESTABLISHED
tcp        0      0 127.0.0.1:50656         127.0.0.1:33220         ESTABLISHED
tcp6       0      0 ::1:25                  :::*                    LISTEN
tcp6       0      0 :::33476                :::*                    LISTEN
tcp6       0      0 ::1:15023               :::*                    LISTEN
```

```
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:22                  ::1:39236               ESTABLISHED
tcp6       0      0 ::1:39236               ::1:22                  ESTABLISHED
udp        0      0 0.0.0.0:32899           0.0.0.0:*
udp        0      0 0.0.0.0:995             0.0.0.0:*
udp        0      0 0.0.0.0:52452           0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 127.0.0.1:613           0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
udp        0      0 0.0.0.0:53110           0.0.0.0:*
udp6       0      0 :::17599                :::*
udp6       0      0 :::995                  :::*
udp6       0      0 :::5353                 :::*
udp6       0      0 :::33524                :::*
udp6       0      0 :::40492                :::*
udp6       0      0 :::111                  :::*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     17625    /tmp//.java_pid1791
--More--
```

Pour connaitre la liste des applications ayant ouvert un port sur l'ordinateur, saisissez la commande suivante :

```
root@debian8:~# netstat -anp | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      868/exim4
tcp        0      0 127.0.0.1:42370         0.0.0.0:*               LISTEN      1791/Remote Access
tcp        0      0 127.0.0.1:15023         0.0.0.0:*               LISTEN      2449/ssh
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      396/rpcbind
tcp        0      0 0.0.0.0:41012           0.0.0.0:*               LISTEN      434/rpc.statd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      471/sshd
```

```
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      4041/inetd
tcp        0      0 127.0.0.1:7127          0.0.0.0:*               LISTEN      1791/Remote Access
tcp        0      0 127.0.0.1:33220         127.0.0.1:50656         ESTABLISHED 1879/Remote Access
tcp        0      0 10.0.2.15:22            10.0.2.2:46432          ESTABLISHED 10584/sshd: trainee
tcp        0      0 127.0.0.1:50656         127.0.0.1:33220         ESTABLISHED 1791/Remote Access
tcp6       0      0 ::1:25                  :::*                    LISTEN      868/exim4
tcp6       0      0 :::33476                :::*                    LISTEN      434/rpc.statd
tcp6       0      0 ::1:15023               :::*                    LISTEN      2449/ssh
tcp6       0      0 :::111                  :::*                    LISTEN      396/rpcbind
tcp6       0      0 :::22                   :::*                    LISTEN      471/sshd
tcp6       0      0 ::1:22                  ::1:39236               ESTABLISHED 2415/sshd: trainee
tcp6       0      0 ::1:39236               ::1:22                  ESTABLISHED 2449/ssh
udp        0      0 0.0.0.0:32899           0.0.0.0:*                           419/dhclient
udp        0      0 0.0.0.0:995             0.0.0.0:*                           396/rpcbind
udp        0      0 0.0.0.0:52452           0.0.0.0:*                           482/avahi-daemon: r
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           482/avahi-daemon: r
udp        0      0 0.0.0.0:68              0.0.0.0:*                           419/dhclient
udp        0      0 127.0.0.1:613           0.0.0.0:*                           434/rpc.statd
udp        0      0 0.0.0.0:111             0.0.0.0:*                           396/rpcbind
udp        0      0 0.0.0.0:53110           0.0.0.0:*                           434/rpc.statd
udp6       0      0 :::17599                :::*                                419/dhclient
udp6       0      0 :::995                  :::*                                396/rpcbind
udp6       0      0 :::5353                 :::*                                482/avahi-daemon: r
udp6       0      0 :::33524                :::*                                482/avahi-daemon: r
udp6       0      0 :::40492                :::*                                434/rpc.statd
udp6       0      0 :::111                  :::*                                396/rpcbind
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name    Path
unix  2      [ ACC ]     STREAM     LISTENING     17625    1791/Remote Access  /tmp//.java_pid1791
--More--
```

Résolution d'adresses Ethernet

Chaque protocole peut être encapsulé dans une **trame** Ethernet. Lorsque la trame doit être transportée de l'expéditeur au destinataire, ce premier doit connaitre l'adresse Ethernet du dernier. L'adresse Ethernet est aussi appelée l'adresse **Physique** ou l'adresse **MAC**.

Pour connaître l'adresse Ethernet du destinataire, l'expéditeur fait appel au protocol **ARP**. Les informations reçues sont stockées dans une table. Pour visualiser ces informations, il convient d'utiliser la commande suivante :

```
root@debian8:~# arp -a
? (10.0.2.2) at 52:54:00:12:35:02 [ether] on eth0
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on eth0
```

Options de la commande

Les options de cette commande sont :

```
root@debian8:~# arp --help
Usage:
  arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]             <-Display ARP cache
  arp [-v]          [-i <if>] -d  <host> [pub]               <-Delete ARP entry
  arp [-vnD] [<HW>] [-i <if>] -f  [<filename>]            <-Add entry from file
  arp [-v]   [<HW>] [-i <if>] -s  <host> <hwaddr> [temp]             <-Add entry
  arp [-v]   [<HW>] [-i <if>] -Ds <host> <if> [netmask <nm>] pub           <-''-

        -a                      display (all) hosts in alternative (BSD) style
        -s, --set               set a new ARP entry
        -d, --delete            delete a specified entry
        -v, --verbose           be verbose
        -n, --numeric           don't resolve names
        -i, --device            specify network interface (e.g. eth0)
        -D, --use-device        read <hwaddr> from given device
        -A, -p, --protocol      specify protocol family
```

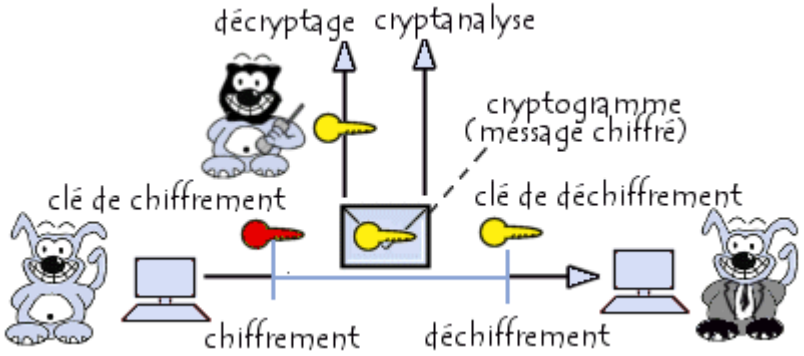
```
       -f, --file              read new entries from file or from /etc/ethers

<HW>=Use '-H <hw>' to specify hardware address type. Default: ether
List of possible hardware types (which support ARP):
  ash (Ash) ether (Ethernet) ax25 (AMPR AX.25)
  netrom (AMPR NET/ROM) rose (AMPR ROSE) arcnet (ARCnet)
  dlci (Frame Relay DLCI) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
  irda (IrLAP) x25 (generic X.25) eui64 (Generic EUI-64)
```

Annexe #3 - Comprendre le Chiffrement

Introduction à la cryptologie

Définitions

- **La Cryptologie**
  - La science qui étudie les aspects scientifiques de ces techniques, c'est-à-dire qu'elle englobe la cryptographie et la cryptanalyse.
- **La Cryptanalyse**
  - Lorsque la clef de déchiffrement n'est pas connue de l'attaquant on parle alors de cryptanalyse ou cryptoanalyse (on entend souvent aussi le terme plus familier de cassage).
- **La Cryptographie**
  - Un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Les verbes crypter et chiffrer sont utilisés.
- **Le Décryptement ou Décryptage**
  - Est le fait d'essayer de déchiffrer illégitimement le message (que la clé de déchiffrement soit connue ou non de l'attaquant).



La Cryptographie

La cryptographie apporte quatre points clefs:

- La confidentialité
  - consiste à rendre l'information inintelligible à d'autres personnes que les acteurs de la transaction.
- L'intégrité
  - consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).
- L'authentification
  - consiste à assurer l'identité d'un utilisateur.
- La non-répudiation
  - est la garantie qu'aucun des correspondants ne pourra nier la transaction.

La cryptographie est basée sur l'arithmétique. Il s'agit, dans le cas d'un texte, de transformer les lettres qui composent le message en une succession de chiffres (sous forme de bits dans le cas de l'informatique), puis ensuite de faire des calculs sur ces chiffres pour:

- Procéder au chiffrement
  - Le résultat de cette modification (le message chiffré) est appelé cryptogramme (Ciphertext) par opposition au message initial, appelé message en clair (Plaintext)
- Procéder au déchiffrement

Le chiffrement se fait à l'aide d'une clef de chiffrement. Le déchiffrement nécessite une clef de déchiffrement.

On distingue deux types de clefs:

- Les clés symétriques:
  - des clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- Les clés asymétriques:
  - des clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

Le Chiffrement par Substitution

Le chiffrement par substitution consiste à remplacer dans un message une ou plusieurs entités (généralement des lettres) par une ou plusieurs autres entités. On distingue généralement plusieurs types de cryptosystèmes par substitution :

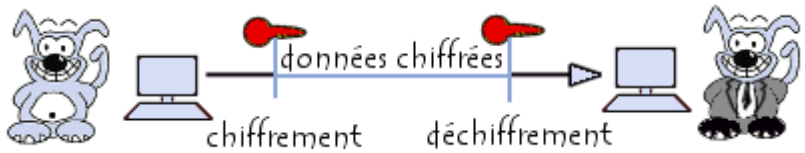
- La substitution **monoalphabétique**
  - consiste à remplacer chaque lettre du message par une autre lettre de l'alphabet
- La substitution **polyalphabétique**
  - consiste à utiliser une suite de chiffres monoalphabétique réutilisée périodiquement
- La substitution **homophonique**
  - permet de faire correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères
- La substitution de **polygrammes**
  - consiste à substituer un groupe de caractères (polygramme) dans le message par un autre groupe de caractères

Algorithmes à clé secrète

Le Chiffrement Symétrique

Ce système est aussi appelé le système à **Clef Secrète** ou à **clef privée**.

Ce système consiste à effectuer une opération de chiffrement par algorithme mais comporte un inconvénient, à savoir qu'il nécessite un canal sécurisé pour la transmission de la clef de chiffrement/déchiffrement.



Le système de Méthode du Masque Jetable (One Time Pad) fût mis au point dans les années 1920. Il utilisait une clef générée aléatoirement à usage unique.

Les algorithmes de chiffrement symétrique couramment utilisés en informatique sont:

- **Data Encryption Standard** (DES),
- **Triple DES** (3DES),
- **RC2**,
- **Blowfish**,
- **International Data Encryption Algorithm** (IDEA),
- **Advanced Encryption Standard** (AES).

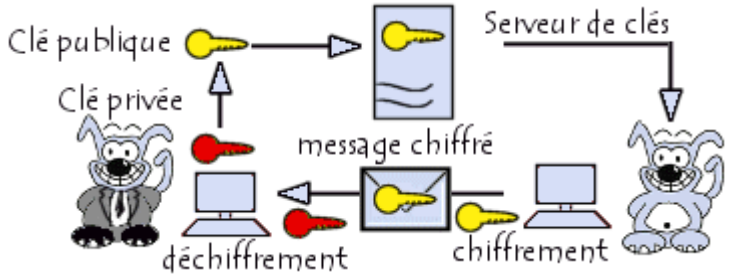
Algorithmes à clef publique

Le Chiffrement Asymétrique

Ce système est aussi appelé **Système à Clef Publique**.

Ce système consiste à avoir deux clefs appelées des **bi-clefs**:

- Une clef **publique** pour le chiffrement
- Une clef **secrète** ou **privée** pour le déchiffrement



- L'utilisateur A (celui qui déchiffre) choisit une clef privée.
- A partir de cette clef il génère plusieurs clefs publiques grâce à un algorithme.
- L'utilisateur B (celui qui chiffre) choisit une des clefs publiques à travers un canal non-sécurisé pour chiffrer les données à l'attention de l'utilisateur A.

Ce système est basé sur ce que l'on appelle une **fonction à trappe à sens unique** ou **one-way trap door**.

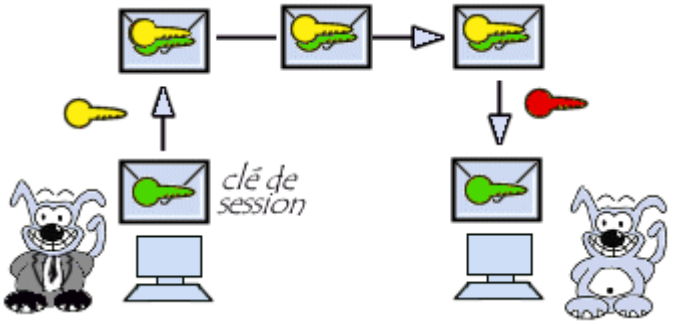
Il existe toutefois un problème – s'assurer que la clef publique récupérée est bien celle qui correspond au destinataire !

Les algorithmes de chiffrement asymétrique couramment utilisés en informatique sont:

- **Digital Signature Algorithm** (DSA)
- **Rivest, Shamir, Adleman** (RSA)

La Clef de Session

Ce système est un compromis entre le système symétrique et le système asymétrique. Il permet l'envoie de données chiffrées à l'aide d'un algorithme de chiffrement symétrique par un canal non-sécurisé et a été mis au point pour palier au problème de lenteur de déchiffrement du système asymétrique.

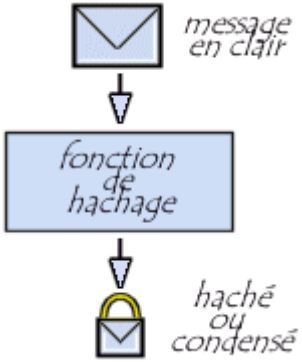


Ce système fonctionne de la façon suivante :

- L'utilisateur A chiffre une clef privée générée aléatoirement, appelée une « clef de session », en utilisant une des clefs publiques de l'utilisateur B.
- L'utilisateur A chiffre les données avec la clef de session.
- L'utilisateur B déchiffre la clef de session en utilisant sa propre clef privée.
- L'utilisateur B déchiffre les données en utilisant la clef de session.

Fonctions de Hachage

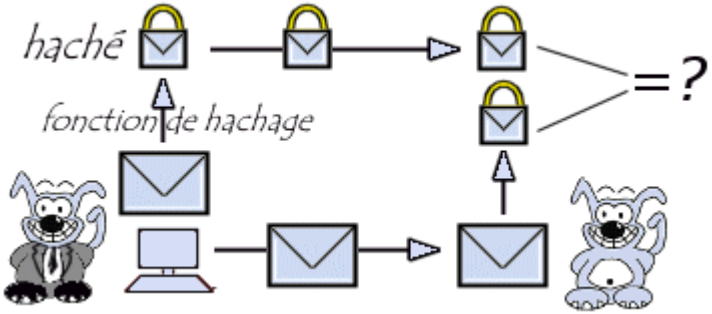
La fonction de **hachage**, aussi appelée une fonction de **condensation**, est à **sens unique** (one way function). Il « condense » un message en clair et produit un haché unique.



Les deux algorithmes de hachage utilisés sont:

- **Message Digest 5** (MD5)
- **Secure Hash Algorithm** (SHA)

Lors de son envoie, le message est accompagné de son haché et il est donc possible de garantir son intégrité:

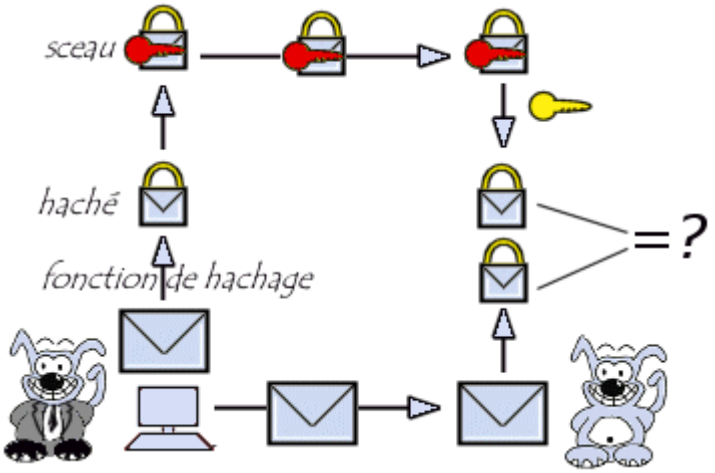


- A la réception du message, le destinataire ou l'utilisateur B calcule le haché du message reçu et le compare avec le haché accompagnant le document.
- Si le message ou le haché a été falsifié durant la communication, les deux empreintes ne correspondront pas.

> Ce système permet de vérifier que l'empreinte correspond bien au message reçu, mais ne permet pas de prouver que le message a bien été envoyé par l'utilisateur A.

Signature Numérique

Pour garantir l'authentification du message l'utilisateur A va chiffrer ou **signer** le haché à l'aide de sa clé privée. Le haché signé est appelé un **sceau**.



- L’utilisateur A envoie le sceau au destinataire.
- A la réception du message L’utilisateur B déchiffre le sceau avec la clé publique de l’utilisateur A.
- Il compare le haché obtenu au haché reçu en pièce jointe.

Ce mécanisme de création de sceau est appelé **scellement**.

Ce mécanisme est identique au procédé utilisé par SSH lors d'une connexion

Utilisation de GnuPG

Présentation

GNU Privacy Guard permet aux utilisateurs de transférer des messages chiffrés et/ou signés.

Installation

Sous RHEL/CentOS 7, le paquet gnupg est installé par défaut :

```
root@debian8:~# whereis gpg
gpg: /usr/bin/gpg /usr/share/man/man1/gpg.1.gz
```

Configuration

Pour initialiser GnuPG, saisissez la commande suivante :

```
root@debian8:~# whereis gpg
gpg: /usr/bin/gpg /usr/share/man/man1/gpg.1.gz
root@debian8:~# gpg
gpg: directory `/root/.gnupg' created
gpg: new configuration file `/root/.gnupg/gpg.conf' created
gpg: WARNING: options in `/root/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/root/.gnupg/secring.gpg' created
gpg: keyring `/root/.gnupg/pubring.gpg' created
gpg: Go ahead and type your message ...
^C
gpg: Interrupt caught ... exiting
```

Pour aider gpg dans la génération des clefs, utilisez **rngd** pour fournir suffisament d”Entropy“ au noyau :

```
root@debian8:~# apt-get install rng-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  rng-tools
0 upgraded, 1 newly installed, 0 to remove and 95 not upgraded.
Need to get 46.8 kB of archives.
After this operation, 209 kB of additional disk space will be used.
Get:1 http://ftp.fr.debian.org/debian/ jessie/main rng-tools amd64 2-unofficial-mt.14-1 [46.8 kB]
Fetched 46.8 kB in 0s (146 kB/s)
```

```
Selecting previously unselected package rng-tools.
(Reading database ... 82472 files and directories currently installed.)
Preparing to unpack .../rng-tools_2-unofficial-mt.14-1_amd64.deb ...
Unpacking rng-tools (2-unofficial-mt.14-1) ...
Processing triggers for systemd (215-17+deb8u4) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up rng-tools (2-unofficial-mt.14-1) ...
Job for rng-tools.service failed. See 'systemctl status rng-tools.service' and 'journalctl -xn' for details.
invoke-rc.d: initscript rng-tools, action "start" failed.
Processing triggers for systemd (215-17+deb8u4) ...

root@debian8:~# rngd -f -r /dev/urandom
rngd 2-unofficial-mt.14 starting up...
entropy feed to the kernel ready
^Cstats: bits received from HRNG source: 60064
stats: bits sent to kernel pool: 4096
stats: entropy added to kernel pool: 4096
stats: FIPS 140-2 successes: 3
stats: FIPS 140-2 failures: 0
stats: FIPS 140-2(2001-10-10) Monobit: 0
stats: FIPS 140-2(2001-10-10) Poker: 0
stats: FIPS 140-2(2001-10-10) Runs: 0
stats: FIPS 140-2(2001-10-10) Long run: 0
stats: FIPS 140-2(2001-10-10) Continuous run: 0
stats: HRNG source speed: (min=64.005; avg=64.949; max=65.998)Mibits/s
stats: FIPS tests speed: (min=99.861; avg=111.541; max=124.663)Mibits/s
stats: Lowest ready-buffers level: 2
stats: Entropy starvations: 0
stats: Time spent starving for entropy: (min=0; avg=0.000; max=0)us
Exiting...

root@debian8:~# cat /proc/sys/kernel/random/entropy_avail
2022
```

Pour générer les clefs, saisissez la commande suivante :

```
root@debian8:~# gpg --gen-key
gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: I2TCH
Email address: infos@i2tch.eu
Comment: Test Key
You selected this USER-ID:
```

```
    "I2TCH (Test Key) <infos@i2tch.eu>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 158 more bytes)
.......+++++
.+++++
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
+++++
..+++++
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 324951F4 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/324951F4 2016-08-07
      Key fingerprint = 293A 4AB0 C917 DEAD 1838  FAE6 B112 5F50 3249 51F4
uid                  I2TCH (Test Key) <infos@i2tch.eu>
sub   2048R/315943DA 2016-08-07
```

La liste de clefs peut être visualisée avec la commande suivante :

```
root@debian8:~# gpg --list-keys
/root/.gnupg/pubring.gpg
------------------------
pub   2048R/324951F4 2016-08-07
uid                  I2TCH (Test Key) <infos@i2tch.eu>
sub   2048R/315943DA 2016-08-07
```

Pour importer la clef d'un correspondant dans sa trousse de clefs il convient d'utiliser la commande suivante :

```
# gpg --import la-clef.asc
```

Pour exporter sa clef publique, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --export --armor I2TCH > ~/I2TCH.asc
root@debian8:~# cat I2TCH.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQENBFem/AUBCADD1ExMqUny634CUdpqY8zvkaaPdg3JdV3dSmUEMOhhqyaftp++
RLqhi7F7+/uSTNv1I611xsVQ1zgPO8YTIwM2c/glIKa6pRTfPFO7qzPczFMhwNrO
a2jnwJjlqg4/BB8N0xWry2TTOjn6lTcwtWcFzjy0jmjFDQLut0/85mnGNB2aml9r
iaSMF2XwvjSIz6nPv0EVvfzzLUdebKBDBrqiZebCbCfeTzR3yyMr9rA8QWdfCHuW
yQaI/FO9BvHdv6CTmBXlKcfu+nTuBfhUfWlWIYn2549Fy47KIdLS+rB7b91FlUw+
8kidI4AVIUm2WlmpLzTJN/0m1PuBoHgi8TLzABEBAAG0IUkyVENIIChUZXN0IEtl
eSkgPGluZm9zQGkydGNoLmV1PokBOAQTAQIAIgUCV6b8BQIbAwYLCQgHAwIGFQgC
CQoLBBYCAwECHgECF4AACgkQsRJfUDJJUfSz3ggAwsH18xuOjidLNST2OM+8Sf/6
954Ajp3x7zBSQCihhHnuVL/vwlrIJ8ScHudQDPq42+zghfH+37O1PXU7cv43hsXT
+cLtSRjGtVE3uvScIUodJONGJJM6oOf3doyLsYHfA1511IWViryTGylS5sBBfjcN
/ljiSjE9K/IRqJ6g1iKU8reC5DU+FmjpOJsnTTef2Je91iHfuR+DPZUzgimulI5G
0Nrq4pqOD/o1j55N+AKM3fqlqRlyUuozsT+CfBGC/xN9gj1FTeeof3bwNoWuiD75
8nYA97eX0jQooxyrHq+9HHU6kvFtOVVpUEbgHyZzenQzdbAuYabJTETE7vgMfbkB
```

```
DQRXpvwFAQgArXPkbmoLS/B7eBUm1cTVvkcJET/RG4AcVs4aZNZ24ve8/qNva7Ec
d4A3kG1t3rHKlFlHnsGm8tHw3Jjg+/6WFFAzG4mzm8QrwA+vnmcHmSrhVgCaA0NS
vqOIWCys96bKcwLIJuYDK9kLuUDRRniMcaA2sl44BaVDl9V8HEm3PS3jYbewwCZa
Z7vtiiK39cyn6AatZHBw7ubeYupmtTUc34dfSym5K7jKgg3VOhaVGsDF1OPogBit
w1tLdNHRyxYkIkhV9xtBIaUMSDsfulmMVsQXXCwY27m3EQMc25C7xuQ5K/+TRLsv
DKzcYcPcHd0cT8B7Ym7+Xetq+CdQD1jOSQARAQABiQEfBBgBAgAJBQJXpvwFAhsM
AAoJELESX1AySVH0/PsIAKwbeV1fwcucq3X+afa/DtEmFWlRS50XwqVGb/ADu10R
7XsftBUSRBPMTvKvFNLSOklggKKP2OFz3ZmBttjdJXL+24U48LtLplG2cfXpAzjD
7rMVuKICgJGHlLWr+sT3t2uH/Mw7Rn3aw0a1MpV5GoqrBvKfdcYbTcbp9HSzCaJL
eVYbXTwXRLlyhcPVy2BZl8OVQlizqLkuzonAmzOMUpYAXU983MZlzEKhMp8R/otC
vx/Y+7kCHiWBqZzpD1MktjeSxUWt+jC7z4pb13t7DOFUzysiBdAYhWtCWAIbJdO2
Pi6GtIZkpKnV1Q7Ct6x3K7HJKfpis3YjIZSFq+Gr09o=
=bhYQ
-----END PGP PUBLIC KEY BLOCK-----
```

Cette clef peut ensuite être jointe à des messages électroniques ou bien déposée sur un serveur de clefs tel http://www.keyserver.net.

Signer un message

Créez maintenant un message à signer :

```
root@debian8:~# vi ~/message.txt
root@debian8:~# cat ~/message.txt
This is a test message for gpg
```

Pour signer ce message en format binaire, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --default-key I2TCH --detach-sign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 324951F4, created 2016-08-07
```

```
root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.sig
□□□□□□□□W□□□
    □□_P2IQ□E□□□>□□)7□Lt□□□□C□+□vC□0□□cX□□□□□1N□Z□6□;□□□□□R□□□□Y□□>□gЧ□□□c□□□□T□iO|&□□TF□□□h□□□□u□□□□□V□
                              _ă□□Tf□□9ct□□□ě□□Q□i>□□ж□□□"K□c□□)□□□~□□□□*□u□f%□□DykheG/u□□_□K□'□□}G□H
                        □R^i□tЛь□y□`□[]a□_1□дP)□a□□\□□□□ЭFO$m□□l
                                                                    3z□□□я□{□Ҫe□*H□□w□H□N   □□□□x□ڳ□□
```

Pour signer ce message en format ascii, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --default-key I2TCH --armor --detach-sign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 324951F4, created 2016-08-07

root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      473 Aug 10 11:29 message.txt.asc
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.asc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAABAgAGBQJXqwICAAoJELESX1AySVH0tqUH/igOKaMwaEIgPCwUpv8rJLIy
2cCj2stliojf437f7eZDyHTNiB/ghmLc6GCogV3uA+jJg2O5vidI5HTD7M2qnnix
CTFRMwH0v61ssZTU/nB+KyO2S3NWRGV1T/dGXJGjXf2QBcSowWoTeQBjbVGDnB3O
twDpWH8NFdW7yGNoKdnPkCdrKEMuUCthLz05W1yCngFDpPwkd8MM9wFq7UdQ+EN9
G2iiyIQbArOI18v67BH571z+4UlOfJlUB+O2C9D8tobJBU7KNSPLWnYAJck6YrrC
```

```
3RVW5M74LcHtfQypSeCJAqxiZf2SMtNU99zPnDqMzwX8tNCpYrlL3IX6DdduOJ4=
=/wL9
-----END PGP SIGNATURE-----
```

Pour vérifier la signature d'un message signé en mode ascii, il convient d'utiliser la commande :

```
root@debian8:~# gpg --verify message.txt.asc
gpg: assuming signed data in `message.txt'
gpg: Signature made Wed 10 Aug 2016 11:34:35 BST using RSA key ID 324951F4
gpg: Good signature from "I2TCH (Test Key) <infos@i2tch.eu>"
```

> Pour vérifier la signature d'un message signé en mode ascii et produit en dehors du message lui-même, il convient d'utiliser la commande :
>
> ```
> # gpg --verify message.txt.asc message.txt
> ```

Pour signer ce message **dans le message lui-même** en format ascii, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --default-key I2TCH --clearsign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 324951F4, created 2016-08-07

File `message.txt.asc' exists. Overwrite? (y/N) y
root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      551 Aug 10 11:30 message.txt.asc
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig
root@debian8:~# cat message.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
```

```
Hash: SHA1

This is a test message for gpg
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAEBAgAGBQJXqwJHAAoJELESX1AySVH0VEMIAJc7NP2v8s/mmllpRi3Kj9W1
oTS721z/XSbM4iQaFM7QnoJmlfCaAbOB01WNqiAVL4A1LGFntttknsoMF7lERU6k
hPCeMtdcWTF3/KwlLHBZ3jjNJeVS4BfJjiW0qStgbMRuzaVHV0iONACA80mnYEOx
TkVnk/IMkO0fsCBocbFeJ/DdR/P9o4u4yqhkwin2+cKPoEWUYBODhIOtHzLMuq1m
582UmrGUQq5z6CC7kiZzifbOtm54pT5MioVfHpYwt6+zlfvYhgVn8VQ62eKAg0zs
IaaRTYVmlD1XUbWxswqvBA9RwIRck6A5Oi5YAoH8jUHaZjvVK9KaEXDQ7Ga/Nk4=
=i5f6
-----END PGP SIGNATURE-----
```

Chiffrer un message

Pour chiffrer un message, il faut disposer de la clef publique du destinataire du message. Ce dernier utilisera ensuite sa clef privée pour déchiffrer le message. Il convient de préciser le destinataire du message, ou plus précisément la clef publique à utiliser, lors d'un chiffrement :

```
  gpg --recipient <destinataire> --encrypt <message>
```

- <*destinataire*> représente toute information permettant de distinguer sans ambigüité une clef publique dans votre trousseau. Cette information peut-être le nom ou l'adresse email associé à la clef publique que vous voulez utiliser,
- <*message*> représente le message à chiffrer.

Par exemple pour chiffrer un message en mode binaire, il convient de saisir la commande suivante :

```
root@debian8:~# gpg --recipient I2TCH --encrypt message.txt

root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      473 Aug 10 11:34 message.txt.asc
```

```
-rw-r--r-- 1 root root      367 Aug 10 11:37 message.txt.gpg
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.sig
□□□□□□□□W□□□
    □□_P2IQ□E□□□>□□)7□Lt□□□□C□+□vC□0□□cX□□□□□1N□Z□6□;□□□□□R□□□□Y□□>□gЧ□□□c□□□□T□iO|&□□TF□□□h□□□□u□□□□□V□
                              _ă□□Tf□□9ct□□□ě□□Q□i>□□ж□□□"K□c□□)□□□~□□□□*□u□f%□□DykheG/u□□_□K□'□□}G□H
                       □R^i□tЈь□y□`□[]a□_1□ժP)□a□□\□□□□ɘFO$m□□l
                                                                  3z□□□я□{□ҷe□*H□□w□H□N  □□□□x□ڰ□□
```

Et pour chiffrer un message en mode ascii, il convient de saisir la commande suivante :

```
root@debian8:~# gpg --recipient I2TCH --armor --encrypt message.txt
File `message.txt.asc' exists. Overwrite? (y/N) y

root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      579 Aug 10 11:38 message.txt.asc
-rw-r--r-- 1 root root      367 Aug 10 11:37 message.txt.gpg
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1

hQEMA9hdNoMxWUPaAQgAk8S4GyQNVeB36uoLY0icuC/WXoxL5P7cJHXVnQuPQiBU
XJX3rHW0RCavc6mLTJvUGYUysubcVisQFWm3LTZ0ZD796S671EdLNNGgsfHNdR90
ext6f6UihR2ep5Y++8eTSat7YQc8nLiF5yexbnOCBuyxGI7gP2lLiZAX6sifmY61
pMegrrhA4BN2Wupbwm2A7WX4NcU4mdZtNxW+zC1Rtp+NOr6ad5JftEes2yPdUcrD
WW2gZaL1DJx50OmBc/tmUt1ea2VEtVDr1WDosD6dEWUbffDBA6wzr1DzUW488D0s
mYoVwu1bYSSSzmNGGvlFDa6EE/nHwVMhvgld1SB63tJeAbYgXKgEyKTfgIe/Byss
EyofLf5p+DVtJs1MK3OdJ87GV5n2XamtD3Qp3O2EvF/YW9e9aRmt6rF9jLSbIBiS
m4Ka3BM8p2yK9PQQbAvQIIyUg9TjP31c1bzdRa/VNQ==
```

```
=laaP
-----END PGP MESSAGE-----
```

Pour décrypter un message il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --decrypt message.txt.asc

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 315943DA, created 2016-08-07 (main key ID 324951F4)

gpg: encrypted with 2048-bit RSA key, ID 315943DA, created 2016-08-07
      "I2TCH (Test Key) <infos@i2tch.eu>"
This is a test message for gpg
```

PKI

On appelle **PKI** (Public Key Infrastucture, ou en français **infrastructure à clé publique (ICP)**, parfois **infrastructure de gestion de clés (IGC)**) l'ensemble des solutions techniques basées sur la cryptographie à clé publique.

Les cryptosystèmes à clés publiques permettent de s'affranchir de la nécessité d'avoir recours systématiquement à un canal sécurisé pour s'échanger les clés. En revanche, la publication de la clé publique à grande échelle doit se faire en toute confiance pour assurer que :

- La clé publique est bien celle de son propriétaire ;
- Le propriétaire de la clé est digne de confiance ;
- La clé est toujours valide.

Ainsi, il est nécessaire d'associer au bi-clé (ensemble clé publique / clé privée) un certificat délivré par un **tiers de confiance** : l'infrastructure de gestion de clés.

Le tiers de confiance est une entité appelée communément autorité de certification (ou en anglais Certification authority, abrégé CA) chargée d'assurer la véracité des informations contenues dans le certificat de clé publique et de sa validité.

Pour ce faire, l'autorité signe le certificat de clé publique à l'aide de sa propre clé en utilisant le principe de signature numérique.

Le rôle de l'infrastructure de clés publiques est multiple et couvre notamment les champs suivants :

- enregistrer des demandes de clés en vérifiant l'identité des demandeurs ;
- générer les paires de clés (clé privée / clé publique) ;
- garantir la confidentialité des clés privées correspondant aux clés publiques ;
- certifier l'association entre chaque utilisateurs et sa clé publique ;
- révoquer des clés (en cas de perte par son propriétaire, d'expiration de sa date de validité ou de compromission).

Une infrastructure à clé publique est en règle générale composée de trois entités distinctes :

- L'autorité d'enregistrement (AE ou RA pour Recording authority), chargée des formalité administratives telles que la vérification de l'identité des demandeurs, le suivi et la gestion des demandes, etc.) ;
- L'autorité de certification (AC ou CA pour Certification Authority), chargée des tâches techniques de création de certificats. L'autorité de certification est ainsi chargée de la signature des demandes de certificat (CSR pour Certificate Signing Request, parfois appelées PKCS#10, nom du format correspondant). L'autorité de certification a également pour mission la signature des listes de révocations (CRL pour Certificate Revocation List) ;
- L'Autorité de dépôt (Repository) dont la mission est de conserver en sécurité les certificats.

Certificats X509

Pour palier aux problèmes liés à des clefs publiques piratées, un système de certificats a été mis en place.

Le certificat permet d’associer la clef publique à une entité ou une personne. Les certificats sont délivrés par des Organismes de Certification.

Les certificats sont des fichiers divisés en deux parties :

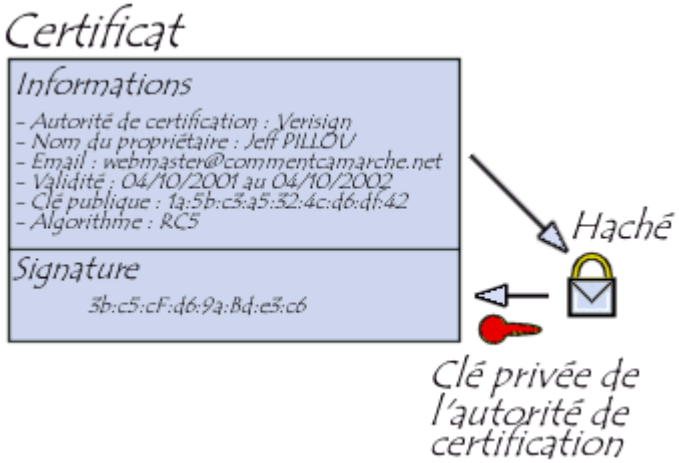
- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification

La structure des certificats est normalisée par le standard **X.509** de l’**Union internationale des télécommunications**.

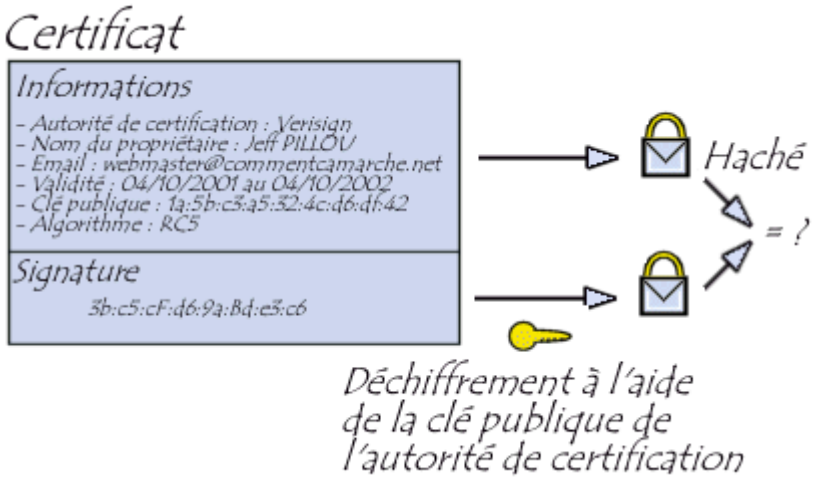
Elle contient :

- Le nom de l'autorité de certification
- Le nom du propriétaire du certificat
- La date de validité du certificat
- L'algorithme de chiffrement utilisé
- La clé publique du propriétaire

Le Certificat est signé par l'autorité de certification:



La vérification se passe ainsi:



<html>

</html>
