Systèmes, réseaux et sécurité

Master Sciences & Technologie
Mention Mathématiques et Sciences Pour l'Ingénieur
Spécialité Informatique
Parcours Ingénierie du Logiciel Libre
2ème Année

Année 2009-2010

D'après la version de Jean-Christophe Soulié (2007-2008)
David DUVIVIER
Laboratoire d'Informatique du Littoral (LIL)
Université du Littoral Côte d'Opale - BP719 62228 Calais – France
50 Rue Ferdinand Buisson – Équipe MESC – Bureau C036
Tél: +33(0)21465675 - E-mail: david.duvivier@lil.univ-littoral.fr

Plan

- FTP / Apache / Logrotate
- DHCP / DNS
- Netfiltering / IP Table
- Squid & SquidGuard
- Port Knocking

Timing

- FTP / Apache – (HTTP) / Logrotate
  - → TP N°1
- DHCP / DNS
  - → TP N°2
- Netfiltering / IP Table
  - → TP N°3
- Squid & SquidGuard / Port Knocking
  - → TP N°4

FTP

Présentation Générale

- Le protocole FTP (File Transfer Protocol) est, comme son nom l'indique, un protocole de transfert de fichier
- La mise en place du protocole FTP date de 1971 (au MIT)
- Le protocole FTP est actuellement défini par la RFC 959 (*cf*. http://rfc.net)

Rôle du protocole FTP

- Le protocole FTP définit la façon selon laquelle des données doivent être transférées sur un réseau TCP/IP
- Le protocole FTP a pour objectifs de :
  - Permettre un partage de fichiers entre machines distantes
  - Permettre une indépendance aux systèmes de fichiers des machines clientes et serveur
  - Permettre de transférer des données de manière efficace

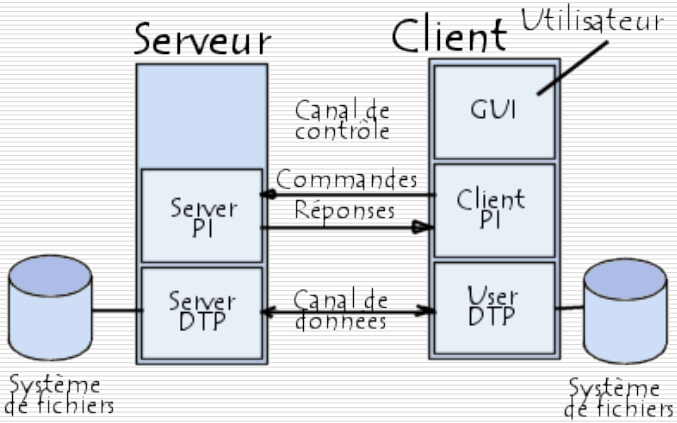
Modèle FTP

- Le protocole FTP s'inscrit dans un modèle client-serveur
- Pendant une connexion FTP, deux canaux de transmission sont ouverts:
  - Un canal pour les commandes (canal de contrôle)
  - Un canal pour les données

Modèle FTP

- le client comme le serveur possèdent deux processus permettant de gérer ces deux types d'information :
  - Le **DTP** (Data Transfer Process) est le processus chargé d'établir la connexion et de gérer le canal de données. Le DTP côté serveur est appelé SERVER-DTP, le DTP côté client est appelé USER-DTP
  - Le **PI** (Protocol Interpreter) est l'interpréteur de protocole permettant de commander le DTP à l'aide des commandes reçues sur le canal de contrôle. Il est différent sur le client et sur le serveur :
  - Le **SERVER-PI** est chargé d'écouter les commandes provenant d'un USER-PI sur le canal de contrôle sur un port donné, d'établir la connexion pour le canal de contrôle, de recevoir sur celui-ci les commandes FTP de l'USER-PI, d'y répondre et de piloter le SERVER-DTP
  - Le **USER-PI** est chargé d'établir la connexion avec le serveur FTP, d'envoyer les commandes FTP, de recevoir les réponses du SERVER-PI et de contrôler le USER-DTP si besoin

Modèle FTP



Vérification de l'activité FTP

ps alx | grep ftp → Liste des processus ftp

netstat –apv | grep ftp → Statistiques ftp (processus…)

Les commandes FTP

- Les commandes FTP permettent de préciser :
  - Le port utilisé
  - Le mode de transfert des données
  - La structure des données
  - La nature de l'action à effectuer (Retrieve, List, Store, ...)

Les commandes FTP

- On distingue trois types de commandes FTP :
  - Les commandes de contrôle d'accès
    - USER, CWD, QUIT, …
  - Les commandes du paramétrage de transfert
    - PORT, TYPE, …
  - Les commandes de service FTP
    - RETR, STOR, DELE, MKD, …

HTTP

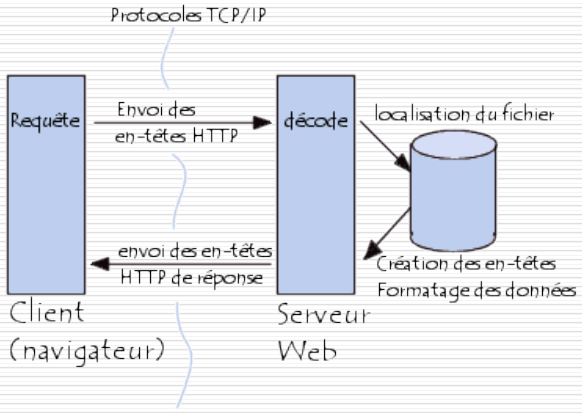
Introduction

- Le protocole HTTP (HyperText Transfer Protocol) est le protocole le plus utilisé sur Internet depuis 1990
- La version 0.9 était uniquement destinée à transférer des données sur Internet
- La version 1.0 du protocole permet de transférer des messages avec des en-têtes décrivant le contenu du message en utilisant un codage de type MIME

Communication entre navigateur et serveur

- La communication entre le navigateur et le serveur se fait en deux temps :
  - Le navigateur effectue une **requête HTTP**
  - Le serveur traite la requête puis envoie une **réponse HTTP**
- Cela peut être un peu plus compliqué (par exemple, si on utilise des CGI)

Communication entre navigateur et serveur



Requête HTTP

- Une requête HTTP est un ensemble de lignes envoyé au serveur par le navigateur qui comprend :
  - **Une ligne de requête**: c'est une ligne précisant le type de document demandé, la méthode qui doit être appliquée, et la version du protocole utilisée. La ligne comprend trois éléments devant être séparés par un espace :
    - La méthode
    - L'URL
    - La version du protocole utilisé par le client (généralement *HTTP/1.0*)

Requête HTTP

  - **Les champs d'en-tête de la requête**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la requête et/ou le client (Navigateur, système d'exploitation, ...)
  - **Le corps de la requête**: c'est un ensemble de lignes optionnelles et permettant, par exemple, un envoi de données par une commande POST lors de l'envoi de données au serveur par un formulaire

Requête HTTP

- Exemple :

```
GET http://www.monurl.pasnet HTTP/1.0

Accept : text/html

If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT

User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
```


Requête HTTP

- Exemple concret N°1 :

```
telnet www-lil.univ-littoral.fr 80
GET /index.html HTTP/1.1
HOST: www-lil.univ-littoral.fr
```

} ← ATTENTION 2 sauts de ligne

Réponse :

```
HTTP/1.1 200 OK
Date : …
Server : … → PLEIN D'INFORMATIONS UTILES ➔ ☺ / ☹
…
Content-Length, Content-Type puis le code HTML

close
```

Requête HTTP

- Exemple concret N°2 :

```
telnet www.ietf.org 80 | tee testhtml.log
GET /rfc.html HTTP/1.1
HOST: www.ietf.org
                                    } ← ATTENTION 2 sauts de ligne

close
emacs testhtml.log &
```


Réponse HTTP

- Une réponse HTTP est un ensemble de lignes envoyées au navigateur par le serveur qui comprend :
  - **Une ligne de statut**: c'est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. La ligne comprend trois éléments devant être séparés par un espace :
    - La version du protocole utilisé
    - Le code de statut
    - La signification du code

Réponse HTTP

- **Les champs d'en-tête de la réponse**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur
- **Le corps de la réponse**: il contient le document demandé

Réponse HTTP

- Exemple :

```
HTTP/1.0 200 OK

Date : Sat, 15 Jan 2000 14:37:12 GMT

Server : Microsoft-IIS/2.0

Content-Type : text/HTML

Content-Length : 1245

Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT
```

Plus d'information

- RFC 1945 - Hypertext Transfer Protocol - HTTP/1.0
  http://www.ietf.org/rfc/rfc1945.txt

- RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1
  http://www.ietf.org/rfc/rfc2616.txt

Logrotate

Description

- **Logrotate** facilite l'administration des systèmes qui génèrent un grand nombre de journaux → typiquement : les serveurs !
- Il automatise la permutation, la compression, la suppression, et l'envoi des journaux
- Chaque journal peut être traité quotidiennement, hebdomadairement, mensuellement, ou quand il devient trop volumineux

Installé, actif ou non ?

- **dpkg –l | grep logrotate**
  pour savoir s'il est installé
- **aptitude install logrotate**
  pour l'installer
- **ls -l /etc/cron.?????*/logrotate** Doit afficher une ligne relative à la fréquence à laquelle le script de lancement de logrotate est invoqué par [ana]cron
  → en principe il est dans /etc/cron.daily
  → *i.e.* lancé une fois par jour

DHCP

Présentation

- DHCP = Dynamic Host Configuration Protocol
- Protocole réseau dont le rôle est d'assurer la configuration automatique des paramètres IP d'une station
  - Par exemple, en lui assignant automatiquement une adresse IP et un masque de sous-réseau
  - Ou en configurant l'adresse de la passerelle par défaut, des serveurs de noms DNS

Pourquoi ?

- Initialement, IP utilisait une configuration statique (adresse IP statique)
- Avec l'agrandissement des réseaux, il y avait des problèmes de maintenance et des risques d'erreurs
- Ou tout simplement, pas assez d'adresses IP à donner
  - Par exemple, les FAI

Pourquoi ?

- DHCP apporte une solution à ces deux inconvénients :
  - Seuls les ordinateurs en service utilisent une adresse de l'espace d'adressage
  - Toute modification des paramètres (adresse de la passerelle, des serveurs de noms) est répercutée sur les stations lors du redémarrage
  - La modification de ces paramètres est centralisée sur les serveurs DHCP

Historique et références

- Le protocole a été présenté pour la première fois en octobre 1993
- Il est défini par la RFC1531
  - Modifié et complété par les RFC 1534, RFC 2131 et RFC 2132
- DHCP peut fonctionner avec IPv4, mais il fonctionne aussi avec IPv6

Fonctionnement côté client

- Les messages DHCP sont transmises via UDP
- L'ordinateur équipé de TCP/IP, mais dépourvu d'adresse IP, envoie par diffusion un datagramme (**DHCP DISCOVER**) qui s'adresse au port 67 de n'importe quel serveur à l'écoute sur ce port
- Ce datagramme comporte entre autres l'adresse physique (MAC) du client

Fonctionnement côté client

- Tout serveur DHCP ayant reçu ce datagramme, s'il est en mesure de proposer une adresse (**DHCP OFFER**) sur le réseau auquel appartient le client, diffuse une offre DHCP à l'attention du client (sur son port 68), identifié par son adresse physique
- Cette offre comporte l'adresse IP du serveur, ainsi que l'adresse IP et le masque de sous-réseau qu'il propose au client. Il se peut que plusieurs offres soient adressées au client

Fonctionnement côté client

- Le client retient une des offres reçues (la première qui lui parvient), et diffuse sur le réseau un datagramme de requête DHCP (**DHCP REQUEST**)
- Ce datagramme comporte l'adresse IP du serveur et celle qui vient d'être proposée au client
- Elle a pour effet de demander au serveur choisi l'assignation de cette adresse, l'envoi éventuel des valeurs des paramètres, et d'informer les autres serveurs qui ont fait une offre qu'elle n'a pas été retenue

Fonctionnement côté client

- Le serveur DHCP choisi élabore un datagramme d'accusé de réception (**DHCP ACK**) qui assigne au client l'adresse IP et son masque de sous-réseau, la durée du bail de cette adresse, deux valeurs T1 et T2 qui déterminent le comportement du client en fin de bail, et éventuellement d'autres paramètres :
  - Adresse IP de la passerelle par défaut
  - Adresses IP des serveurs DNS
  - Adresses IP des serveurs NBNS (WINS, dans le monde Windows)

Fonctionnement côté client - Bail

- Les adresses IP dynamiques sont octroyées pour une durée limitée, qui est transmise au client dans l'accusé de réception qui clôture la transaction DHCP
- La valeur T1 qui l'accompagne détermine la durée après laquelle le client commence à demander périodiquement le renouvellement de son bail auprès du serveur qui lui a accordé son adresse (couramment la moitié de la durée du bail)
- Cette fois la transaction est effectuée par transmission IP classique, d'adresse à adresse

Fonctionnement côté client - Bail

- Si lorsque le délai fixé par la deuxième valeur, T2, est écoulé, le bail n'a pas pu être renouvelé (par exemple si le serveur DHCP d'origine est hors service), le client demande une nouvelle allocation d'adresse par diffusion
- Si au terme du bail le client n'a pu ni en obtenir le renouvellement, ni obtenir une nouvelle allocation, l'adresse est désactivée et il perd la faculté d'utiliser le réseau TCP/IP de façon normale

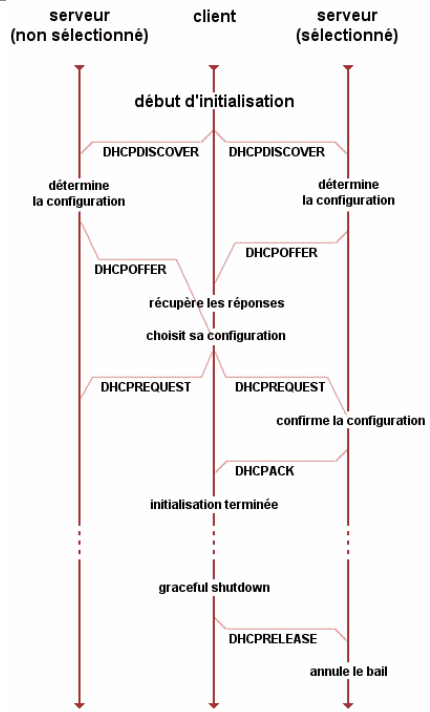
Fonctionnement côté client - Bail

- On peut optimiser l'attribution des adresses IP en jouant sur la durée des baux
- Exemple : si aucune adresse n'est libérée au bout d'un certain temps, plus aucune requête DHCP ne pourra être satisfaite, faute d'adresses à distribuer
- Sur un réseau où beaucoup d'ordinateurs se branchent et se débranchent souvent (réseau d'école ou de locaux commerciaux par exemple), il est intéressant de proposer des baux de courte durée
- A l'inverse, sur un réseau constitué en majorité de machines fixes, très peu souvent rebootées, des baux de longues durées suffisent

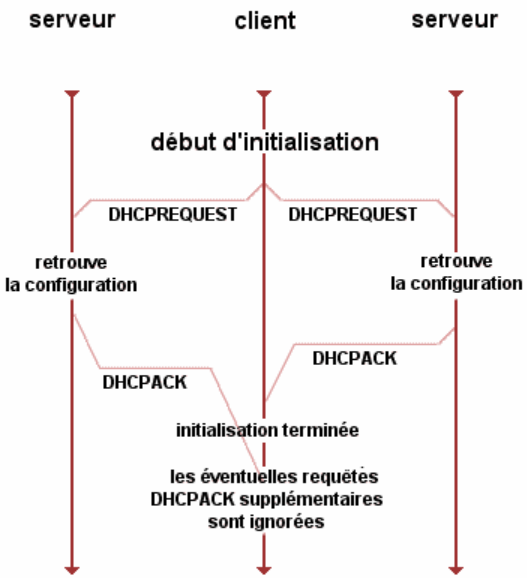
Fonctionnement côté client – pour résumer

- **DHCPDISCOVER** (pour localiser les serveurs DHCP disponibles)
- **DHCPOFFER** (réponse du serveur à un paquet **DHCPDISCOVER**, qui contient les premiers paramètres)
- **DHCPREQUEST** (requête diverse du client pour par exemple prolonger son bail)
- **DHCPACK** (réponse du serveur qui contient des paramètres et l'adresse IP du client)
- **DHCPNAK** (réponse du serveur pour signaler au client que son bail est échu ou si le client annonce une mauvaise configuration réseau)
- **DHCPDECLINE** (le client annonce au serveur que l'adresse est déjà utilisée)
- **DHCPRELEASE** (le client libère son adresse IP)
- **DHCPINFORM** (le client demande des paramètres locaux, il a déjà son adresse IP)

Demande de nouvelle adresse



Renouvellement d'une adresse



Fonctionnement côté client – Un peu plus dans le détail

- La trame DHCP est en fait la même que BOOTP, et a le format suivant

octet 1	octet 2	octet 3	octet 4
op (1)	htype (1)	hlen (1)	hops (1)
xid (4)			
secs (2)		flags (2)	
ciaddr (4)			
yiaddr (4)			
siaddr (4)			
giaddr (4)			
chaddr (16)			
sname (64)			
file (128)			
options (variable)			

Fonctionnement côté client – Un peu plus dans le détail

- **op** : vaut 1 pour BOOTREQUEST (requête client), 2 pour BOOTREPLY (réponse serveur)
- **htype** : type de l'adresse hardware (adresse MAC, par exemple)
- **hlen** : longueur de l'adresse hardware (en octet). C'est 6 pour une adresse MAC
- **hops** : peut être utilisé par des relais DHCP
- **xid** : nombre aléatoire choisi par le client et qui est utilisé pour reconnaître le client
- **secs** : le temps écoulé (en secondes) depuis que le client a commencé sa requête
- **flags** : flags divers
- **ciaddr** : adresse IP du client, lorsqu'il en a déjà une

Fonctionnement côté client – Un peu plus dans le détail

- **yiaddr** : la (future ?) adresse IP du client
- **siaddr** : adresse IP du (prochain) serveur à utiliser
- **giaddr** : adresse IP du relais (passerelle par exemple) lorsque la connexion directe client/serveur n'est pas possible
- **chaddr** : adresse hardware du client
- **sname** : champ optionnel. Nom du serveur
- **ile** : nom du fichier à utiliser pour le boot
- **options** : Champs réservé Rfc 2132. Dans une précédente RFC, la taille de ce champ était limitée (limité à 64 octets par exemple pour la première version de Bootp) ; maintenant, il n'y a plus de limitation.
- Dans tous les cas, un client DHCP doit être prêt à recevoir au minimum 576 octets, mais la possibilité lui est offerte de demander au serveur de restreindre la taille de ses messages

Fonctionnement côté client – Un peu plus dans le détail

- Le passage de paramètres (nom de la machine...) se fait par l'intermédiaires d'options
- Les options sont documentées dans la Rfc 2132
- Elles portent toutes un numéro qui les identifie. Par exemple, l'option 15 est celle qui permet de donner au client le nom de domaine du réseau
- Il est possible d'envoyer plusieurs options dans le même message DHCP
- Dans tous les cas, que l'on ne transmette qu'une seule option utile ou plusieurs, on doit toujours finir la zone d'options par une option 255 (end)

Fonctionnement côté client – Un peu plus dans le détail

- Le format des options est le suivant :

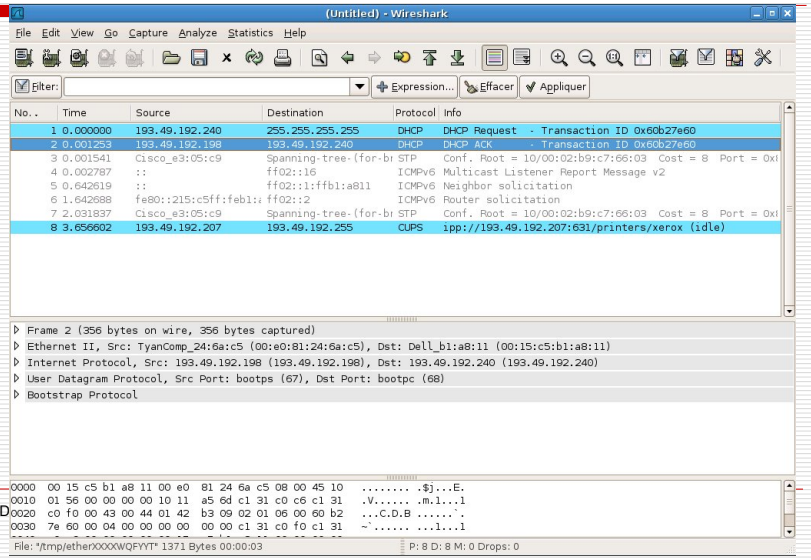
octet 1	octet 2	données...
Code de l'option	longeur champ de données	...

- Le numéro de l'option n'est codé que sur 1 octet (donc il ne peut y avoir que 256 options possibles)
- L'octet 2 code la longueur du champ de données qui suit
- Il ne tient donc pas compte des 2 octets de code d'option et de longueur

Fonctionnement côté client – Un peu plus dans le détail

- Certaines options ne comportent pas de données complémentaires, comme l'option 255
- Dans ce cas, il n'y a ni champ de longueur ni champ de données
- Les messages DHCP comme DHCPACK, ... sont des options
- Il s'agit de l'option 53 qui comporte un champ de données de longueur 1 contenant le numéro identifiant la requête (1 pour DHCPDISCOVER, par exemple).

Fonctionnement côté client – Un peu plus dans le détail



Fonctionnement côté serveur

- Le serveur s'appelle **`dhcp server`**
  → On lance une recherche avec ces mots-clefs (dhcp et server) dans le cache local des paquets debian:

```
# apt-cache search dhcp server
...
dhcp3-server - DHCP server for automatic IP address assignment
...
dhcp3-relay - DHCP relay
...
 # apt-get install dhcp3-server
```


Fonctionnement côté serveur

- Le paramétrage du serveur dhcp se fait par le fichier « /etc/dhcp3/dhcpd.conf » sous Debian
- Par défaut, il se trouve dans le répertoire /etc/, mais vous pouvez le mettre n'importe où.
- Il est composé de plusieurs sections, chacune limitée par des accolades { et } :
  - des paramètres globaux qui s'appliquent à tout le fichier
  - shared-network,
  - subnet,
  - host,
  - group.

Fonctionnement côté serveur

- Chaque section peut contenir des paramètres et des options
- Une section group peut contenir des sections host.
- Au début du fichier, on peut placer des paramètres globaux, comme par exemple la durée des baux, les adresses des DNS...
- Chaque ligne du fichier de configuration doit se terminer par un « ; » sauf lorsqu'il y a une accolade
- Les commentaires sont possibles en ajoutant un # en début de ligne

Fonctionnement côté serveur

- Ils peuvent être un peu tout et n'importe quoi, pourvu qu'ils aient une signification applicable à toutes les déclarations du fichier
- Par exemple, on peut redéfinir la durée des baux (max-lease-time et default-lease-time), empêcher le serveur de répondre à des requêtes venant d'hôtes non déclarés (deny unknown-clients;), indiquer le nom de domaine que les machines doivent utiliser, les serveurs DNS...

Fonctionnement côté serveur

- shared-network
  - Ce paramètre est utilisé pour regrouper plusieurs zones subnet lorsque ceux-ci concerne le même réseau physique
  - Les paramètres rentrés en début de zone seront utilisés pour le boot des clients provenant des sous-réseaux déclarés, à moins de spécifier pour certains hôtes de ne pas booter (zone host)
  - Son utilisation se rapproche de celle de host ; il faut toutefois l'utiliser systématiquement que le réseau est divisé en différents sous-réseaux administrés par le serveur DHCP

Fonctionnement côté serveur

- shared-network

Syntaxe :

```
shared-network FOO-BAR
{
filename "boot";

subnet 192.168.2.0 netmask 255.255.255.224
{
range 192.168.2.10 192.168.2.30;
}
subnet 192.168.2.32 netmask 255.255.255.224
{
range 192.168.2.40 192.168.2.50;
}

}
```

Fonctionnement côté serveur

- subnet
  - permet de définir les sous-réseaux sur lesquels le serveur DHCP doit intervenir
  - C'est la partie la plus importante du fichier de configuration du serveur DHCP
  - Sans ça, le serveur DHCP ne marchera jamais

Fonctionnement côté serveur

- subnet

Syntaxe :

```
subnet numero_sous-reseau netmask
netmask
{
[ paramètres globaux... ]
[ déclarations... ]
}
```


Fonctionnement côté serveur

- subnet
  - numero_sous-reseau et netmask sont donnés sous format adresse IP pointées
  - On peut bien entendu commencer la zone par des paramètres globaux qui ne seront appliqués que pour les ordinateurs de ce sous-réseau
    - Par exemple, le nom de domaine à appliquer sur ce sous-réseau (option domain-name)

Fonctionnement côté serveur

- host
  - Ce mot permet de déclarer des machines que le DHCP doit connaître et leur appliquer une configuration particulière
  - On est pas obligé d'utiliser cette zone, mais elle est, par exemple, indispensable lorsque l'on a déclaré deny unknown-clients; en début de fichier pour empêcher le serveur DHCP de répondre à des requêtes provenant d'hôtes non déclarés

Fonctionnement côté serveur

- host

Syntaxe :

```
host nom
{
paramètres...
}
```


Fonctionnement côté serveur

- host
  - Un hôte peut être reconnu de deux façons :
    - En utilisant son nom (le nom qui suit le mot clé host) ou en utilisant son adresse hardware (ethernet ou token-ring)
    - Dans ce dernier cas, il faut ajouter une ligne dans la déclaration host : hardware ethernet| token-ring adresse-hardware;
    - Il est fortement recommandé d'authentifier les ordinateurs à partir de leur adresse hardware plutôt que leur nom, surtout qu'il sont supposés ne pas posséder de véritable nom Internet et que l'on peut redéfinir ce nom

Fonctionnement côté serveur

- host
  - Un point important : c'est dans une déclaration host que l'on décide d'attribuer une adresse fixe ou non à un hôte
  - Il suffit alors d'utiliser une ligne comme celle-ci : fixed-address 192.168.2.4;
  - ATTENTION ! Toute adresse IP attribuée de manière fixe ne doit pas faire partie des zones d'adresses IP déclarées avec range... (zone subnet)

Fonctionnement côté serveur

- group
  - Cette zone est simplement utilisée pour rassembler plusieurs déclarations (de toute sorte, y compris d'autres déclarations group) pour leur appliquer des différents paramètres

Fonctionnement côté serveur

- group

Syntaxe :

```
group
{
option domain-name "bar.org";
option routers 192.168.1.254;

host foo1
{
...
}

host foo2
{
...
}

}
```


Fonctionnement côté serveur

- Les options
  - Les paramètres qui doivent commencer avec option sont les options définies dans la Rfc 2132
  - Il y en a environ 60 définies dans la Rfc, mais le serveur peut en gérer jusqu'à 254 (les options 0 et 255 sont réservées)
  - Pour trouver les options possibles et leur nom, on peut consulter le fichier common/tables.c des sources du serveur
  - **ATTENTION** ! les noms des options peut varier d'une version de serveur à une autre.

Fonctionnement côté serveur

- Les options
  - Le format des valeurs des options est donné dans ce même fichier au début ("format codes:"). Les options plus utiles sont les suivantes :
    - **- subnet-mask** (option 1) qui indique le masque de sous-réseau pour la configuration IP
    - **- routers** (option 3) qui indique les routeurs à utiliser
    - **- domain-name-servers** (option 6) qui indique les DNS à utiliser
    - **- host-name** (option 12) qui indique au client quel nom d'hôte il doit prendre
    - **- domain-name** (option 15) qui fournit au client le nom du domaine arpa dans lequel il se trouve
    - **- broadcast-address** (option 28) qui indique l'adresse de broadcast en vigueur sur le réseau
    - **- dhcp-lease-time** (option 51) qui indique au client la durée de validité du bail
    - - D'autres options (60 en particulier) permettent de personnaliser les messages DHCP circulant sur le réseau


```
- max-lease-time 240;
- default-lease-time 240;
- deny unknown-clients;
- option domain-name "bar.com";
- option domain-name-servers foo1.bar.com, foo2.bar.com;

subnet 192.168.1.0 netmask 255.255.255.0
{
range 192.168.1.2 192.168.1.100;
range 192.168.1.110 192.168.1.254;
option broadcast-address 192.168.1.255;
}
group
{
option routers 192.168.2.101;

host foo3
{
hardware ethernet 00:c0:c3:11:90:23;
option host-name pc3;
}

host foo4
{
hardware ethernet 00:c0:c3:cc:0a:8f;
fixed-address 192.168.1.105;
}

}
```

```
group
{
option routers 192.168.2.101;

host foo3
{
hardware ethernet 00:c0:c3:11:90:23;
option host-name pc3;
}

host foo4
{
hardware ethernet 00:c0:c3:cc:0a:8f;
fixed-address 192.168.1.105;
}

}

host foo5
{
hardware ethernet 00:c0:c3:2a:34:f5;
server-name "bootp.bar.com";
filename "boot";
}
```

Fonctionnement côté serveur

- Les cinq premières lignes définissent les paramètres globaux
  - Les 2 premiers concernent les baux (leases)
  - La ligne suivante dit au serveur de ne pas répondre aux requêtes DHCP venant d'hôtes qu'il ne connaît pas (i.e. non définis dans dhcpd.conf)
  - On définit enfin les paramètres du domaine du réseau (nom de domaine et serveurs DNS).
- On définit ensuite le sous-réseau sur lequel le serveur DHCP est censé intervenir : c'est la ligne "subnet..."
- Dans ce sous-réseau, on dit au serveur de ne fournir des adresses IP que dans les plages d'adresses définies par les lignes « range…»
- La dernière ligne de la section définit l'adresse de broadcast à utiliser sur le sous-réseau

Fonctionnement côté serveur

- On crée ensuite un groupe dont le but est uniquement de fournir des adresses de passerelles à des machines bien déterminées (par leur adresse MAC)
- On remarque que foo4.bar.com obtiendra une adresse IP fixe
- foo5, enfin, sera une machine qui bootera à travers le réseau, en se connectant au serveur TFTP bootp.bar.com, et booter avec le fichier boot.

Fonctionnement côté serveur – Un autre exemple

```
#On donne le nom du domaine
option domain-name "nom_choisi";
#On défini le masque réseau
option subnet-mask 255.255.255.0;
# Ici c'est le serveur de nom, le serveur privé,
# il faut aussi mettre le/les DNS
option domain-name-servers 192.168.1.2 , 192.168.1.3;
ddns-update-style ad-hoc;
# Les clients auront cette adresse comme passerelle par défaut
option routers 192.168.1.1;
#Le bail a une durée de 86400 s par défaut, soit 24 h
default-lease-time 86400;
#On le laisse avec un maximum de 7 jours
max-lease-time 604800;
# Définition du réseau : 192.168.1.0 et de son masque
subnet 192.168.1.0 netmask 255.255.255.0 {
#La plage d'adresses disponibles pour les clients
range 192.168.1.4 192.168.1.253;
# Et l'adresse utilisée pour la diffusion
option broadcast-address 192.168.1.255;
}
```


Fonctionnement côté serveur

- Base de données d'attribution
  - Sur le serveur DHCP, le fichier **`/var/lib/dhcp/dhcpd.leases`** stocke la base de données d'attribution client DHCP
  - Les informations d'attribution DHCP pour toutes les adresses IP récemment attribuées sont automatiquement stockées dans cette base de données
  - Ces informations comprennent la durée de l'attribution, le destinataire de l'attribution d'adresse IP, les dates de début et de fin pour l'attribution et l'adresse MAC de la carte d'interface réseau qui a été utilisée pour l'attribution
  - La première fois qu'on lance le service DHCP, celui-ci ne fonctionnera pas si la base de données d'attribution n'existe pas

Fonctionnement côté serveur

- Base de données d'attribution
  - Il faut utiliser la commande **`touch /var/lib/dhcpd.leases`** pour créer le fichier avant de lancer le serveur pour la première fois
  - Une fois que le fichier existe et que le serveur a été lancé, il ne faut pas essayer de créer un nouveau fichier de base de données d'attribution
  - La base de données d'attribution est recréée de temps en temps de façon à ce que sa taille ne soit pas trop importante
  - Tout d'abord, toutes les attributions connues sont sauvegardées dans une base de données d'attribution temporaire
  - Le fichier **`dhcpd.leases`** est renommé **`dhcpd.leases~`**, et la base de données d'attribution temporaire est copiée dans **`dhcpd.leases`**

Fonctionnement côté serveur

- Exemple de lease :

```
lease 192.168.12.58
   starts 2 1999/08/24 06:28:48;
   ends 3 1999/08/25 06:28:48;
   hardware ethernet 00:10:5a:2e:56:a7;
   uid 01:00:10:5a:2e:56:a7;
   client-hostname "Toto";
```


Fonctionnement côté serveur

- Exemple :

No.	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0x6719436e
2	0.001182	192.168.0.253	192.168.0.9	ICMP	Echo (ping) request
3	0.342454	192.168.0.253	192.168.0.9	DHCP	DHCP Offer - Transaction ID 0x6719436e
4	0.344405	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x6719436e
5	0.348264	192.168.0.253	192.168.0.9	DHCP	DHCP ACK - Transaction ID 0x6719436e
6	0.353014	CIS_b9:49:37	Broadcast	ARP	Who has 192.168.0.9? Tell 192.168.0.9
7	0.571241	CIS_b9:49:37	Broadcast	ARP	Who has 192.168.0.9? Tell 192.168.0.9
8	1.571441	CIS_b9:49:37	Broadcast	ARP	Who has 192.168.0.9? Tell 192.168.0.9
9	2.580537	192.168.0.9	192.168.0.255	NBNS	Registration NB PCHRIS<00>
10	2.590265	192.168.0.9	192.168.0.255	NBNS	Registration NB PCHRIS<03>

Fonctionnement côté serveur – Exemple

1. Notre client se réveille, il n'a pas d'IP et utilise 0.0.0.0 pour faire un "broadcast général (255.255.255.255)". C'est le DHCP Discover
2. Notre serveur DHCP, qui a l'intention d'offrir à ce client l'IP 192.168.0.9, fait un ping sur cette adresse, histoire de voir si elle est réellement disponible sur le réseau.
3. Comme il ne reçoit pas de réponse à son ping, il offre cette adresse au client.
4. Le client fait alors un DHCP Request
5. Le serveur accepte
6. Le client fait un broadcast ARP pour vérifier de son côté que l'IP 192.168.0.9 n'est pas dupliquée sur le réseau.
7. idem
8. idem
9. ...

DNS – Présentation Générale

Qu'appelle-t-on DNS ?

- Chaque ordinateur directement connecté à internet possède au moins une adresse IP propre (fixe ou dynamique)
- Cependant, les utilisateurs ne veulent pas travailler avec des adresses numériques mais plutôt avec des adresse plus explicites

Qu'appelle-t-on DNS ?

- Ainsi, il est possible d'associer des noms en langage courant aux adresses numériques grâce à un système appelé **DNS** (*Domain Name System*)
- On appelle *résolution de noms de domaines* (ou *résolution d'adresses*) la corrélation entre les adresses IP et le nom de domaine associé

Noms d'hôtes

- Aux origines de TCP/IP, étant donné que les réseaux étaient très peu étendus ou autrement dit que le nombre d'ordinateurs connectés à un même réseau était faible, les administrateurs réseau créaient des fichiers appelés tables de conversion manuelle
- Ces tables de conversion manuelle étaient des fichiers séquentiels, généralement nommés hosts ou hosts.txt, associant sur chaque ligne l'adresse IP de la machine et le nom littéral associé, appelé nom d'hôte

Introduction au DNS

- Le système précédent de tables de conversion nécessitait néanmoins la mise à jour manuelle des tables de tous les ordinateurs en cas d'ajout ou de modification d'un nom de machine
- Ainsi, avec l'explosion de la taille des réseaux, et de leur interconnexion, il a fallu mettre en place un système de gestion des noms hiérarchisé et plus facilement administrable
- Le système a été mis au point en novembre 1983 par Paul Mockapetris (RFC 882 et RFC 883), puis révisé en 1987 dans les RFCs 1034 et 1035
- Le DNS a fait l'objet depuis de nombreuses RFCs

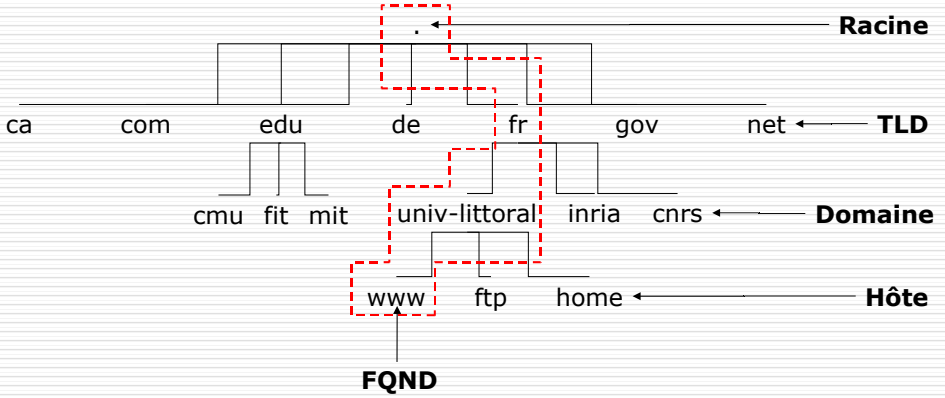
Introduction au DNS

- Ce système propose :
  - Un espace de noms hiérarchique permettant de garantir l'unicité d'un nom dans une structure arborescente, à la manière des systèmes de fichiers d'Unix
  - Un système de serveurs distribués permettant de rendre disponible l'espace de noms
  - un système de clients permettant de «résoudre» les noms de domaines, c'est-à-dire interroger les serveurs afin de connaître l'adresse IP correspondant à un nom

L'espace de noms

- La structuration du système DNS s'appuie sur une structure arborescente dans laquelle sont définis des domaines de niveau supérieurs (appelés **TLD**, pour *Top Level Domains*), rattachés à un noeud racine représenté par un point

L'espace de noms



L'espace de noms

- On appelle « **nom de domaine** » chaque nœud de l'arbre
- Chaque nœud possède une étiquette d'une longueur maximale de 63 caractères
- L'ensemble des noms de domaine constitue ainsi un arbre inversé où chaque nœud est séparé du suivant par un point (« . »)

L'espace de noms

- L'extrémité d'une branche est appelée **hôte**, et correspond à une machine ou une entité du réseau
- Le nom d'hôte qui lui est attribué doit être unique dans le domaine considéré, ou le cas échéant dans le sous-domaine
- Le mot « **domaine** » correspond formellement au suffixe d'un nom de domaine, c'est-à-dire l'ensemble des étiquettes de nœuds d'une arborescence, à l'exception de l'hôte

L'espace de noms

- Le nom absolu correspondant à l'ensemble des étiquettes des nœuds d'une arborescence, séparées par des points, et terminé par un point final, est appelé **adresse FQDN** (*Fully Qualified Domain Name*)
- La profondeur maximale de l'arborescence est de 127 niveaux et la longueur maximale d'un nom FQDN est de 255 caractères
- L'adresse FQDN permet de repérer de façon unique une machine sur le réseau des réseaux
- Par exemple : **`dpt-info.univ-littoral.fr`**

Les serveurs de noms

- Les machines appelées *serveurs de nom de domaine* permettent d'établir la correspondance entre le nom de domaine et l'adresse IP des machines d'un réseau
- Chaque domaine possède un serveur de noms de domaines, appelé « serveur de noms primaire » (*primary domain name server*), ainsi qu'un serveur de noms secondaire (*secondary domaine name server*), permettant de prendre le relais du serveur de noms primaire en cas d'indisponibilité

Les serveurs de noms

- Chaque serveur de nom est déclaré dans à un serveur de nom de domaine de niveau immédiatement supérieur
- Le système de nom est une architecture distribuée, où chaque entité est responsable de la gestion de son nom de domaine
- Il n'existe donc pas d'organisme ayant à charge la gestion de l'ensemble des noms de domaines

Les serveurs de noms

- Les serveurs correspondant aux domaines de plus haut niveau (TLD) sont appelés « **serveurs de noms racine** »
- Il en existe treize, répartis sur la planète, possédant les noms « **`a.root-servers.net`** » à « **`m.root-servers.net`** »

Les serveurs de noms

- Un serveur de noms définit une zone, c'est-à-dire un ensemble de domaines sur lequel le serveur a autorité
- Le système de *noms de domaine* est transparent pour l'utilisateur
- Néanmoins :
  - Chaque ordinateur doit être configuré avec l'adresse d'une machine capable de transformer n'importe quel nom en une adresse IP
  - Cette machine est appelée Domain Name Server
  - L'adresse IP d'un second *Domain Name Server* (secondary Domain Name Server) doit également être définie
- Le serveur le plus répandu s'appelle **BIND** (*Berkeley Internet Name Domain*) maintenu par l'*ISC* (*Internet Systems Consortium*)

Résolution de noms de domaine

- Le mécanisme consistant à trouver l'adresse IP correspondant au nom d'un hôte est appelé « **résolution de nom de domaine** »
- Lorsqu'une application souhaite se connecter à un hôte connu par son nom de domaine, celle-ci va interroger un serveur de noms défini dans sa configuration réseau

Résolution de noms de domaine

- Une requête est ainsi envoyée au premier serveur de noms
- Si celui-ci possède l'enregistrement dans son cache, il l'envoie à l'application, dans le cas contraire il interroge un serveur racine
- Le serveur de nom racine renvoie une liste de serveurs de noms faisant autorité sur le domaine
- Le serveur de noms primaire faisant autorité sur le domaine va alors être interrogé et retourner l'enregistrement correspondant à l'hôte sur le domaine

Types d'enregistrements

- Un DNS est une base de données répartie contenant des enregistrements, appelés **RR** (*Resource Records*)
- En raison du système de cache permettant au système DNS d'être réparti, les enregistrements de chaque domaine possèdent une durée de vie, appelée **TTL** (*Time To Live*)

Types d'enregistrements

- D'une manière générale, un enregistrement DNS comporte les informations suivantes :

FQDN	TTL	Type	Classe	RData
www.yahoo.fr.	3600	A	IN	192.168.22.209

- **Nom de domaine** : le nom de domaine doit être un nom FQDN, c'est-à-dire être terminé par un point

Types d'enregistrements

- **Type** : une valeur sur 16 bits spécifiant le type de ressource décrit par l'enregistrement. Le type de ressource peut être un des suivants :
  - **A** : il s'agit du type de base établissant la correspondance entre un nom canonique et une adresse IP. Par ailleurs il peut exister plusieurs enregistrements A, correspondant aux différentes machines du réseau (serveurs).
  - **CNAME** (*Canonical Name*) : il permet de faire correspondre un alias au nom canonique. Il est particulièrement utile pour fournir des noms alternatifs correspondant aux différents services d'une même machine.
  - **HINFO** : il s'agit d'un champ uniquement descriptif permettant de décrire notamment le matériel (CPU) et le système d'exploitation (OS) d'un hôte. Il est généralement conseillé de ne pas le renseigner afin de ne pas fournir d'éléments d'informations pouvant se révéler utiles pour des pirates informatiques.

Types d'enregistrements

- **MX** (*Mail eXchange*) : correspond au serveur de gestion du courrier. Lorsqu'un utilisateur envoie un courrier électronique à une adresse (utilisateur@domaine), le serveur de courrier sortant interroge le serveur de nom ayant autorité sur le domaine afin d'obtenir l'enregistrement MX. Il peut exister plusieurs MX par domaine, afin de fournir une redondance en cas de panne du serveur de messagerie principal. Ainsi l'enregistrement MX permet de définir une priorité avec une valeur pouvant aller de 0 à 65 535 :

```
www.yahoo.fr. IN     MX     10     mail.yahoo.fr.
```

- **NS** : correspond au serveur de noms ayant autorité sur le domaine.
- **PTR** : un pointeur vers une autre partie de l'espace de noms de domaines.
- **SOA** (*Start Of Authority*) : le champ SOA permet de décrire le serveur de nom ayant autorité sur la zone, ainsi que l'adresse électronique du contact technique (dont le caractère « @ » est remplacé par un point).

Types d'enregistrements

- **Classe** : la classe peut être soit **IN** (correspondant aux protocoles d'internet, il s'agit donc du système utilisé dans notre cas), soit **CH** (pour le système chaotique)
- **RDATA** : il s'agit des données correspondant à l'enregistrement
  - Les informations attendues selon le type d'enregistrement sont :
    - A : une adresse IP sur 32 bits
    - CNAME : un nom de domaine
    - MX : une valeur de priorité sur 16 bits, suivi d'un nom d'hôte
    - NS : un nom d'hôte
    - PTR : un nom de domaine
    - SOA : plusieurs champs

Domaines de haut niveau

- Il existe deux catégories de **TLD** (*Top Level Domain*, soit *domaines de plus haut niveau*) :
  - Les domaines dits « génériques », appelés **gTLD** (*generic TLD*)
  - Les domaines dits «nationaux », appelés **ccTLD** (country code TLD)
    - Les ccTLD correspondent aux différents pays et leurs noms correspondent aux abréviations des noms de pays définies par la norme ISO 3166

Domaines de haut niveau

- Les gTLD historiques :
  - **.arpa** correspond aux machines issues du réseau originel
  - **.com** correspondait initialement aux entreprises à vocation commerciale
    - Désormais ce TLD est devenu le « TLD par défaut » et l'acquisition de domaines possédant cette extension est possible, y compris par des particuliers
  - **.edu** correspond aux organismes éducatifs
  - **.gov** correspond aux organismes gouvernementaux
  - **.int** correspond aux organisations internationales
  - **.mil** correspond aux organismes militaires
  - **.net** correspondait initialement aux organismes ayant trait aux réseaux
    - L'acquisition de domaines possédant cette extension est possible, y compris par des particuliers.
  - **.org** correspond habituellement aux entreprises à but non lucratif

Domaines de haut niveau

- Nouveaux gTLD introduits en novembre 2000 par l'ICANN :
  - **.aero** correspond à l'industrie aéronautique
  - **.biz** (*business*) correspondant aux entreprises commerciales
  - **.museum** correspond aux musées
  - **.name** correspond aux noms de personnes ou aux noms de personnages imaginaires
  - **.info** correspond aux organisations ayant trait à l'information
  - **.coop** correspondant aux coopératives
  - **.pro** correspondant aux professions libérales

Domaines de haut niveau

- gTLD spéciaux :
  - **.arpa** correspond aux infrastructures de gestion du réseau. Le gTLD arpa sert ainsi à la résolution inverse des machines du réseau, permettant de trouver le nom correspondant à une adresse IP

DNS – Jouons un peu avec nslookup !

nslookup

```
D:\>nslookup
Serveur par défaut :  dsldevice.lan
Address:  192.168.1.254
```

On démarre NSLOOKUP en mode interactif.
Le DNS par défaut est celui qui est spécifié dans les options du protocole TCP/IP
Ici le DNS de mon FAI

```
> server g.root-servers.net
Serveur par défaut :  g.root-servers.net
Address:  192.112.36.4
```

Cette commande signifie à nslookup d'utiliser le serveur spécifié,
ici l'un des "root-servers" choisi au hasard.

```
> set q=ns
```

Cette commande indique à nslookup que l'on va s'intéresser aux champs de type "NS" (Name Servers)

```
> fr.
Serveur :  g.root-servers.net
Address:  192.112.36.4

fr      nameserver = A.NIC.fr
fr      nameserver = C.NIC.fr
fr      nameserver = B.EXT.NIC.fr
fr      nameserver = C.EXT.NIC.fr
fr      nameserver = B.NIC.fr
fr      nameserver = D.EXT.NIC.fr
fr      nameserver = A.EXT.NIC.fr
fr      nameserver = E.EXT.NIC.fr
fr      nameserver = E.NIC.fr
A.EXT.NIC.fr    internet address = 193.51.208.14
A.NIC.fr        internet address = 192.93.0.1
A.NIC.fr        AAAA IPv6 address = 2001:660:3005:1::1:1
B.EXT.NIC.fr    internet address = 192.228.90.21
B.NIC.fr        internet address = 192.93.0.4
B.NIC.fr        AAAA IPv6 address = 2001:660:3005:1::1:2
C.EXT.NIC.fr    internet address = 128.112.129.15
C.NIC.fr        internet address = 192.134.0.49
C.NIC.fr        AAAA IPv6 address = 2001:660:3006:1::1:1
D.EXT.NIC.fr    internet address = 204.152.184.85
D.EXT.NIC.fr    AAAA IPv6 address = 2001:4f8:0:2::8
E.EXT.NIC.fr    internet address = 193.176.144.6
E.NIC.fr        internet address = 194.57.253.1
```

Posons la question:
Quels sont les serveurs de noms qui savent donner des informations sur le TLD "fr."?
Et voilà, nous disposons de la liste des serveurs qui pourront nous renseigner sur les domaines existant dans le TLD "fr."


```
> server e.ext.nic.fr
Serveur par défaut :  e.ext.nic.fr
Served by:
- B.ext.nic.fr
        192.228.90.21
        fr
- D.ext.nic.fr
        204.152.184.85
        fr
- E.nic.fr
        194.57.253.1
        fr
- C.ext.nic.fr
        128.112.129.15
        fr
- C.nic.fr
        192.134.0.49
        fr
- B.nic.fr
        192.93.0.4
        fr
- A.ext.nic.fr
        193.51.208.14
        fr
- A.nic.fr
        192.93.0.1
        fr
- e.ext.nic.fr
        193.176.144.6
        fr
```

Passons sur l'un de ces serveurs…

nslookup

```
> univ-reunion.fr.
Serveur :  e.ext.nic.fr
Served by:
…

univ-reunion.fr nameserver = runcache.univ-reunion.fr
univ-reunion.fr nameserver = gate.cru.fr
univ-reunion.fr nameserver = helios.univ-reunion.fr
univ-reunion.fr nameserver = soleil.uvsq.fr
gate.cru.fr     internet address = 195.220.94.33
helios.univ-reunion.fr  internet address = 194.199.73.1
soleil.uvsq.fr  internet address = 193.51.24.1
runcache.univ-reunion.fr        internet address = 195.220.151.50
>
```

Et interrogeons-le sur le domaine univ-reunion.fr.
Nous obtenons les DNS qui nous renseigneront sur les hôtes du domaine univ-reunion.fr

nslookup – Recherche de l'adresse d'un hôte

- Le DNS par défaut

```
D:\>nslookup www.univ-littoral.fr
Serveur :  dsldevice.lan
Address:  192.168.1.254

Réponse ne faisant pas autorité :
Nom :    vador.univ-littoral.fr
Address:  193.49.202.11
Aliases:  www.univ-littoral.fr
```

On obtient donc le nom réel de la machine qui fait office de serveur web, son adresse IP et les alias

nslookup – Recherche de l'adresse d'un hôte

Passons maintenant sur l'un des serveurs DNS du domaine univ-reunion.fr

```
D:\>nslookup
Serveur par défaut :  dsldevice.lan
Address:  192.168.1.254

> server runcache.univ-reunion.fr
Serveur par défaut :  runcache.univ-reunion.fr
Address:  195.220.151.50
```

Le type de question est maintenant: donner l'adresse de...

```
> set q=a
```

Et la réponse arrive

```
> www.univ-littoral.fr.
Serveur :  runcache.univ-reunion.fr
Address:  195.220.151.50

Réponse ne faisant pas autorité :
Nom :    vador.univ-littoral.fr
Address:  193.49.202.11
Aliases:  www.univ-littoral.fr
```



```
D:\>nslookup
Serveur par défaut :  dsldevice.lan
Address:  192.168.1.254

> server runcache.univ-reunion.fr
Serveur par défaut :  runcache.univ-reunion.fr
Address:  195.220.151.50

> ls -d univ-reunion.fr.
...
salazie             A    194.199.73.47
salazie             MX   0    smtp.univ-reunion.fr
satan               A    194.199.73.97
satan               MX   0    smtp.univ-reunion.fr
scfp            CNAME    www.univ-reunion.fr
sciences            CNAME    www.univ-reunion.fr
sciences-dev        CNAME    www.univ-reunion.fr
sget            A    194.199.72.10
sirius              A    194.199.73.6
sirius              MX   0    smtp.univ-reunion.fr
...
 webmail                A    194.199.72.35
 webmail                MX   0    smtp.univ-reunion.fr
 www                CNAME    frontal-http.univ-reunion.fr
...
```

Cette commande permet de lister tous les enregistrements d'une zone (domaine ou partie d'un domaine, comme nous le verrons dans la construction d'un DNS).
Le listing a été volontairement tronqué, d'abord parce qu'il est assez long

nslookup – Recherche de l'adresse d'un hôte

- Pour mémoire :
  - **SOA** (Start Of Authority) Ce champ indique que le serveur concerné (runcache.univ-reunion.fr) est le responsable de la zone univ-reunion.fr (la référence suprême en quelque sorte)
  - **A** (Address) indique l'adresse IP correspondant à un nom d'hôte dans la zone
  - **NS** (Name Server) indique que l'hôte en question est un serveur de noms. On retrouve bien les deux serveurs de noms déjà vus plus haut
  - **CNAME** (Canonical Name) C'est la définition d'un alias. La partie de droite n'est d'ailleurs pas une adresse, mais un nom d'hôte déjà défini dans un enregistrement de type A
  - Il existe également d'autres types d'enregistrements, comme **MX** qui renseigne sur les serveurs smtp

DNS – En route pour la construction d'un DNS !

Un Serveur de Noms qui ne sert que de cache

- Un serveur de noms qui ne sert que de cache trouve la réponse aux requêtes de résolution de nom
- Il se souvient de cette réponse chaque fois qu'on lui posera la même question par la suite
- Cela réduira les temps de réponse, surtout si on a une connexion plutôt lente

Un Serveur de Noms qui ne sert que de cache

- On a tout d'abord besoin du fichier /etc/bind/named.conf ce fichier inclut plusieurs fichiers tels que **named.conf.options, named.conf.local, named.conf.default-zones**
- Le fichier named.conf est lu au lancement de named (*cf.* man named) et provoque la lecture des fichiers inclus
- Commencons par le fichier named.conf.options :

```
// Fichier de config pour un serveur de noms qui ne fait que du cache
options {
        directory "/var/cache/bind";
        // Enlever les commentaires peut vous aider si vous avez a passer a
        // travers un firewall et que ça ne marche pas :
        // query-source port 53;
        // Si votre provider fournit une ou plusieurs adresses IP fixes pour le(s) DNS
        // decommentez ce qui suit et remplacez par la ou les adresses des DNS
        // par exemple en remplaçant 0.0.0.0; par 212.27.40.241; 212.27.40.240;
        // forwarders {
        //         0.0.0.0;
        // };
        auth-nxdomain no;
        listen-on-v6 { any; };
};
```


Un Serveur de Noms qui ne sert que de cache

- La ligne ''directory'' indique à named l'endroit où il doit rechercher ses fichiers
- Certains fichiers dont nous parlerons maintenant auront un chemin relatif à ce répertoire
- Ainsi, si vous ne précisez pas de chemin absolu, pz sera un sous-répertoire de /var/cache/bin, c'est-à-dire se trouver effectivement dans le répertoire /var/cache/bind/pz

Un Serveur de Noms qui ne sert que de cache

- Pour l'instant, le fichier named.conf.local ne contient que des commentaires mentionnant un fichier "/etc/bind/zones.rfc1918". Éditez ce fichier sans le modifier. Il contient les différentes zones relatives aux adresses IP privées (10.X.X.X; 172.16-31.X.X; 192.168.X.X). Si vous n'utilisez aucune de ces zones, décommentez la ligne suivante : //include "/etc/bind/zones.rfc1918". Attention, ce n'est pas notre cas puisque nous utilisons 172.16.X.X, ou dans mon cas 172.16.0.X (sur votre machine le « 0 » est remplacé par le dernier champ de votre adresse IP).
- Il faut intégrer notre propre fichier d'enregistrements DNS relatif à notre domaine « m2i2l.org ». Ce fichier sera le suivant : /etc/bind/db.m2i2l.org. Nous y reviendrons plus tard...

Un Serveur de Noms qui ne sert que de cache

- Insérons les lignes suivantes dans le fichier **named.conf.local** :

```
//zone "0.16.172.in-addr.arpa" { notify no; type master; file
"/etc/bind/db.m2i2l.org"; };
zone "m2i2l.org" { notify no; type master; file "/etc/bind/db.m2i2l.org"; };
```

- Je suppose que la ligne suivante a été ajoutée en fin de fichier **/etc/networks** :

```
m2i2l.org      172.16.0.0
```

- Ces deux lignes sont synonymes en réalité. Nous mentionnons « notify no; » pour ne pas propager les modifications de notre configuration DNS aux autres DNS car il ne faut pas le faire dans le cadre du cours SRS. Si vous configurez un « véritable DNS » qui doit s'insérer parmi les DNS existants, il ne faut pas indiquer « notify no; » dans votre configuration.

Un Serveur de Noms qui ne sert que de cache

- Continuons avec le fichier named.conf.default-zones :

```
// prime the server with knowledge of the root servers
zone "." {
      type hint;
      file "/etc/bind/db.root";
};
// be authoritative for the localhost forward and reverse zones, and for broadcast zones as per RFC 1912
zone "localhost" {
      type master;
      file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
      type master;
      file "/etc/bind/db.127";
};
zone "0.in-addr.arpa" {
      type master;
      file "/etc/bind/db.0";
};
zone "255.in-addr.arpa" {
      type master;
      file "/etc/bind/db.255";
};
```

- A priori nous ne devons pas modifier le fichier named.conf.default-zones; remarquez le fichier db.root -> **.../...**

Un Serveur de Noms qui ne sert que de cache

```
.                    3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.  3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.  3600000      AAAA  2001:503:BA3E::2:30
.                    3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.  3600000      A     192.228.79.201
.                    3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.  3600000      A     192.33.4.12
.                    3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.  3600000      A     128.8.10.90
.                    3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.  3600000      A     192.203.230.10
.                    3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.  3600000      A     192.5.5.241
F.ROOT-SERVERS.NET.  3600000      AAAA  2001:500:2F::F
.                    3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.  3600000      A     192.112.36.4
.                    3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.  3600000      A     128.63.2.53
H.ROOT-SERVERS.NET.  3600000      AAAA  2001:500:1::803F:235
.                    3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.  3600000      A     192.36.148.17
.                    3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.  3600000      A     192.58.128.30
J.ROOT-SERVERS.NET.  3600000      AAAA  2001:503:C27::2:30
.                    3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.  3600000      A     193.0.14.129
K.ROOT-SERVERS.NET.  3600000      AAAA  2001:7FD::1
.                    3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.  3600000      A     199.7.83.42
L.ROOT-SERVERS.NET.  3600000      AAAA  2001:500:3::42
.                    3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.  3600000      A     202.12.27.33
M.ROOT-SERVERS.NET.  3600000      AAAA  2001:DC3::35
```

Un Serveur de Noms qui ne sert que de cache

- Ce fichier donne une description de tous les serveurs de noms du monde qui se trouvent à la racine (au plus haut niveau) de la hiérarchie des serveurs de noms
- Il arrive que cette liste change, c'est pourquoi il est essentiel que ce fichier soit maintenu à jour

Un Serveur de Noms qui ne sert que de cache

- La façon la plus simple de le faire est d'utiliser dig
- On lancez d'abord dig sans argument ou avec « . » comme argument, nous obtenons une version mise à jour du fichier db.root
- Effectivement ce que l'on obtient ressemble énormément à un fichier db.root, avec quelques chiffres en plus
- Ces chiffres supplémentaires sont inoffensifs
- Sauvons-le dans un fichier (dig . >db.root.new) et remplaçons l'ancien fichier db.root avec ce nouveau fichier.
- Posons alors la même question à un des serveurs de cette liste avec (par exemple) la commande dig @e.root-servers.net

Un Serveur de Noms qui ne sert que de cache

- Créons un fichier appelé **db.m2i2l.org** dans le répertoire **/etc/bind** :

```
; Fichier d'enregistrements DNS/BIND pour m2i2l.org
$TTL                    604800
@                       IN  SOA     ns1.m2i2l.org. root.m2i2l.org. (
                2009100702          ; Serial
                  604800            ; Refresh
                   86400            ; Retry
                 2419200            ; Expire
                  604800 )          ; Negative Cache TTL

@                       IN  NS      ns1.m2i2l.org.
ns1.m2i2l.org.          IN  A       172.16.0.1
m2i2l.org.              IN  A       172.16.0.1
www.m2i2l.org.          IN  CNAME   m2i2l.org.
ftp.m2i2l.org.          IN  CNAME   m2i2l.org.
1                       IN  PTR     www.m2i2l.org.
1                       IN  PTR     ftp.m2i2l.org.
1                       IN  PTR     ns1.m2i2l.org.
```

Notes : le numéro de série est la date yyyymmddaa suivie d'un numéro de révision qui doit être incrémenté manuellement à chaque modification de la configuration. Les durées sont en secondes par défaut mail il est possible de préciser des heures (H) des jours (D)...

Un Serveur de Noms qui ne sert que de cache

- Attention : le fichier db.m2i2l.org présenté en slide précédent, n'est pas nécessairement correct (!).
  En ce cas, il vous appartient de le corriger !
- Après ça, nous avons besoin d'un fichier /etc/resolv.conf. Pour l'instant, il peut ressembler à ceci :

```
nameserver 212.27.40.241
nameserver 212.27.40.240
search adsl.proxad.net
```

- Nous voulons obtenir une configuration du genre de celle-ci :

```
nameserver 212.27.40.241
nameserver 212.27.40.240
nameserver 172.16.0.1
search adsl.proxad.net m2i2l.org
```


Un Serveur de Noms qui ne sert que de cache

- La ligne search spécifie dans quels domaines il faudra chercher lorsqu'on voudra se connecter sur une machine de nom quelconque
- La ligne ''nameserver'' indique à quelle adresse notre machine peut contacter un serveur de noms
- Si on veut indiquer plusieurs serveurs de nom, on met une ligne ''nameserver'' pour chacun
- Dans notre cas, il s'agit de notre propre machine puisque c'est elle qui fait tourner named
- named ne lit jamais ce fichier, c'est le *résolveur* qui utilise named qui le fait

Un Serveur de Noms qui ne sert que de cache

- Si nous supposons un instant que le fichier /etc/resolv.conf est le suivant :

```
search subdomain.your-domain.edu your-domain.edu
nameserver 127.0.0.1
```

- Voyons sur un exemple à quoi sert ce fichier :
  - Si un client cherche à contacter foo, on essaye d'abord foo.subdomain.your-domain.edu puis foo.your-domain.edu et enfin foo
  - Si un client essaye de contacter sunsite.unc.edu, on essaye d'abord sunsite.unc.edu.subdomain.your-domain.edu (je sais, c'est stupide, mais c'est comme ça) puis sunsite.unc.edu.your-domain.edu et enfin sunsite.unc.edu
  - Faites attention à ne pas mettre trop de noms de domaine dans la ligne search car cela prend du temps de tous les essayer.

Un Serveur de Noms qui ne sert que de cache

- Cet exemple suppose qu'on appartient au domaine subdomain.your-domain.edu
- Notre machine s'appelle alors certainement your-machine.subdomain.your-domain.edu
- La ligne search ne doit pas contenir notre TLD (Top Level Domain; edu dans ce cas)
- Si on se connecte fréquemment à des machines dans un autre domaine, on peut rajouter ce domaine dans la ligne search comme ceci :

```
search subdomain.your-domain.edu your-domain.edu other-domain.com
```


Un Serveur de Noms qui ne sert que de cache

- Le « problème » sous debian est que pour faciliter la tâche des adminsys, le fichier resolv.conf est automatiquement généré au démarrage par l'outil « resolvconf » (cf. man resolvconf). Pour ajouter les lignes relatives à notre domaine, il faut taper ceci :

```
resolvconf -a  m2i2l.org <<EOF
nameserver 172.16.0.1
search m2i2l.org
EOF
```

- Vérifiez les modifications, via cat /etc/resolv.conf par exemple. Attention, cette modification n'est pas persistante (ce n'est pas plus mal pour l'instant car en cas de problème les erreurs ne survivent pas au restart/reboot !)
- Pour la rendre permanente, il faut modifier le fichier de configuration de resolvconf en ajoutant les deux lignes (**nameserver et search**) dans le fichier **/etc/resolvconf/resolv.conf.d/base**
- « resolvconf -d m2i2l.org » supprime notre ajout temporaire...

Un Serveur de Noms qui ne sert que de cache

- Il est possible de se simplifier la vie en complétant les informations relatives aux dns et domaines en les précisant lors de la configuration des interfaces réseau dans le fichier /etc/network/interfaces (ici j'indique juste ce qui concerne eth0:0) :

```
...
auto eth0:0
iface eth0:0 inet static
  address 172.16.0.1
  netmask 255.255.255.0
  gateway <votre_passerelle>
  dns-nameservers 172.16.0.1
  dns-search m2i2l.org
...
```

- Stoppez l'interface virtuelle eth0:0 via ifdown eth0:0 ou ifconfig eth0:0 down
- Redémarrez l'interface virtuelle via ifup eth0:0

Un Serveur de Noms qui ne sert que de cache

- /etc/host.conf :

```
order hosts,bind
multi on
```

- Si il n'y a pas de ligne ''order'', il faut en mettre une
- Elle indique aux routines de résolution de nom de regarder d'abord dans /etc/hosts puis de demander au serveur de noms (que nous avons précisé dans resolv.conf via resolvconf comme étant 172.16.0.1).

Un Serveur de Noms qui ne sert que de cache

- Dans les grands réseaux, bien administrés, des universités ou FAI (Fournisseur d'Accès a Internet), on peut remarquer que les administrateurs réseau ont mis en place une hiérarchie de serveurs DNS ce qui permet de soulager le réseau interne ainsi que le réseau vers l'extérieur
- Il n'est pas facile de savoir si on est dans un réseau de ce type
- Tout cela n'est pas très important, mais en utilisant le serveur DNS de votre FAI comme ''forwarder'' on peut rendre les réponses plus rapides et alléger la charge de son réseau
- Pour améliorer encore notre exemple, supposons que notre FAI aie deux serveurs de noms qu'il veux vous faire utiliser, ayant pour adresses IP 10.0.0.1 et 10.1.0.1
- Alors, dans notre fichier named.conf, dans la section appelée ''options'' on insère les lignes :

```
forward first;
 forwarders {
 10.0.0.1;
 10.1.0.1;
 };
```


Notre propre domaine

- Maintenant, nous en sommes à définir notre propre domaine bien à nous
- Nous allons créer le domaine m2i2l.org et y déclarer quelques machines
- C'est un nom de domaine totalement factice, afin d'être sûr de ne déranger personne dans le Vaste Monde.

Notre propre domaine

- Tous les caractères ne sont pas admis dans les noms de machines
- On ne doit utiliser que les caractères de l'alphabet anglais (a-z), les nombres (0-9) et le tiret ''-''
- On utilise ces caractères, majuscules et minuscules confondues, donc pat.uio.no est identique à Pat.UiO.No.

Notre propre domaine

- En fait, nous avons déjà commencé à créer notre propre domaine avec cette ligne dans named.conf.local :

```
zone "0.16.172.in-addr.arpa" {
 notify no;
 type master;
 file "/etc/bind/db.m2i2l.org";
};
```

- Ou encore :

```
zone "m2i2l.org" IN {
 notify no;
 type master;
 file "/etc/bind/db.m2i2l.org";
};
```


Notre propre domaine

- Notez bien l'absence de ''.'' à la fin des noms de domaine de ce fichier
- Elle signifie que nous allons définir la zone 0.16.172.in-addr.arpa, que nous sommes son serveur principal et que tout est stocké dans un fichier appelé /etc/bind/db.m2i2l.org
- On a déjà vu ce fichier, il se présente comme ceci :

```
@          IN     SOA    ns1.m2i2l.org hostmaster.m2i2l.org (
 2009100701      ; Serial
 8H     ; Refresh
 2H     ; Retry
 1W     ; Expire
 1D)    ; Minimum TTL
 NS     ns.m2i2l.org.
1                PTR    www.m2i2l.org.
```


Notre propre domaine

- Notons bien le ''.'' à la fin de tous les noms de domaine complets de ce fichier, contrairement au fichier named.boot.
- L'origine (l'emplacement dans la hiérarchie du service DNS) d'un fichier de zone est indiquée dans la zone section du fichier named.conf
- Dans notre cas, c'est 0.16.172.in-addr.arpa.
- Ce ''fichier de zone'' (''zone file''), contient 3 ''resource records'' (RRs) : un SOA RR, un NS RR et un PTR RR
- Le ''@'' est une notation spéciale qui désigne l'origine
- Et comme la colonne ''domain'' de ce fichier donne 0.16.172.in-addr.arpa, la première ligne signifie donc :
  - 0.16.172.IN-ADDR.ARPA. IN SOA ...
- NS est le ''resource records'' pour le serveur de noms (NS = Name Server)

Notre propre domaine

- Elle dit au service DNS quelle machine est le serveur de noms pour le domaine 0.16.172.in-addr.arpa, c'est m2i2l.org
- ns est le nom habituel des serveurs de noms, tout comme www. pour les serveurs Web, mais c'est simplement une habitude, on peut choisir n'importe quel nom
- Et finalement le PTR dit que l'adresse 1 dans le sous réseau 0.16.172.in-addr.arpa, donc 172.16.0.1 est appelé www.m2i2l.org
- Le champ SOA est le préambule de *tous* les fichiers de zone, et il doit y en avoir exactement un dans chaque fichier de zone
- Ce champ SOA décrit la zone, son origine (une machine appelée ns1.m2i2l.org), qui est responsable de son contenu, de quelle version du fichier de zone il s'agit (serial : 2009100701), et quelques autres paramètres pour le cache et les serveurs DNS secondaires. Quant aux champs restants (*refresh*, *retry*, *expire* et *minimum*) on peut se référer à la FAQ

Notre propre domaine

- Maintenant, pour le sujet qui nous préoccupe, le domaine m2i2l.org, insérons une nouvelle zone dans le fichier named.conf.local :

```
zone "m2i2l.org" IN {
 notify no;
 type master;
 file "/etc/bind/db.m2i2l.org";
};
```


Notre propre domaine

- Voici un exemple de fichier de zone linux.bogus (db.linux.bogus), dont vous pouvez vous inspirer :

```
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
 199802151      ; serial, todays date + todays serial #
 8H             ; refresh, seconds
 2H             ; retry, seconds
 1W             ; expire, seconds
 1D )           ; minimum, seconds
;
 TXT    "Linux.Bogus, your DNS consultants"
 NS     ns          ; Inet Address of name server
 NS     ns.friend.bogus.
 MX     10 mail         ; Primary Mail Exchanger
 MX     20 mail.friend.bogus. ; Secondary Mail Exchanger
localhost       A       127.0.0.1
gw              A       192.168.196.1
 HINFO  "Cisco" "IOS"
 TXT    "The router"
ns              A       192.168.196.2
 MX     10 mail
 MX     20 mail.friend.bogus.
 HINFO  "Pentium" "Linux 2.0"
www             CNAME   ns
donald          A       192.168.196.3
 MX     10 mail
 MX     20 mail.friend.bogus.
 HINFO  "i486"  "Linux 2.0"
 TXT    "DEK"
mail            A       192.168.196.4
 MX     10 mail
 MX     20 mail.friend.bogus.
 HINFO  "386sx" "Linux 1.2"
ftp             A       192.168.196.5
 MX     10 mail
 MX     20 mail.friend.bogus.
 HINFO  "P6" "Linux 2.1.86"
```


Notre propre domaine

- Il y a deux choses à noter à propos du champ SOA :
  - ns.linux.bogus *doit absolument* être une vraie machine possédant un champ A
    - Il n'est pas légal d'avoir un champ CNAME pour la machine mentionnée dans le champ SOA. Il n'est pas nécessaire que son nom soit ''ns'', ce peut être tout autre nom valide
  - La deuxième chose à noter c'est que hostmaster.linux.bogus doit se lire comme hostmaster@linux.bogus
    - Ce doit être un alias de mail, ou une véritable boîte aux lettres électronique, et la personne qui maintient le DNS doit la lire régulièrement
    - Tous les mails concernant l'administration du domaine seront envoyés à cette adresse
    - Il n'est pas obligatoire que le nom soit ''hostmaster'', on peut mettre notre adresse e-mail personnelle, mais il serait bon que l'adresse ''hostmaster'' fonctionne aussi.

Notre propre domaine

- Il y a un nouveau RR (Resource Record) dans ce fichier, c'est le MX, pour Mail eXchanger
- Il indique aux systèmes de gestion du courrier électronique à quelle machine envoyer le mail adressé à someone@linux.bogus, dans notre cas à mail.linux.bogus ou mail.friend.bogus
- Le nombre devant chaque machine est sa priorité vis-à-vis du champ MX, le RR avec le numéro le plus faible (10) correspond à la machine à laquelle le courrier doit être adressé en priorité
- En cas d'échec, il peut être adressé à la machine qui a le numéro de priorité immédiatement supérieur, c'est-à-dire mail.friend.bogus qui a une priorité de 20 dans notre cas

Notre propre domaine

- Il y a un certain nombre de nouveaux RR que nous allons passer en revue :
  - HINFO (Host INFOrmation), qui est en deux parties, et c'est une bonne habitude à prendre que d'encadrer chacune de guillemets.
    - La première partie est la description matérielle ou le type de processeur de la machine
    - La deuxième partie décrit le logiciel utilisé ou le système d'exploitation de la machine. ns a pour processeur un Pentium et tourne sous Linux 2.0. Le champ CNAME (Canonical NAME) sert à donner plusieurs noms à la même machine. Par conséquent, www est un alias de ns.
- L'utilisation des champs CNAME est assez controversée. Mais il est sage de suivre la règle selon laquelle un champ MX, CNAME ou SOA ne doit *jamais* se référer à un champ CNAME, toujours se référer à un champ A, il est donc préférable de ne pas avoir :

```
foobar      CNAME   www             ; NON !
#mais plutôt
foobar      CNAME   ns              ; Oui !
```

Notre propre domaine

- Il est aussi important de noter qu'un CNAME n'est pas un nom d'hôte légal pour une adresse de courrier électronique
- webmaster@www.linux.bogus est une adresse de mail illégale avec la configuration ci-dessus
- On peut être sûrs qu'il y a un certain nombre d'administrateurs système dans le Vaste Monde qui sont très à cheval sur cette règle, même si avec un CNAME ça marche pour vous
- Une façon de contourner le problème est d'utiliser des champs A (et peut-être d'autres, comme un champ MX par exemple) à la place :

```
www         A       192.168.196.2
```


Notre propre domaine

- Zone inversée :
  - On a besoin d'une zone inversée pour que l'on puisse retrouver le DNS à partir de l'adresse
  - Ce nom est utilisé par différents types de serveurs (FTP, IRC, WWW et autres) pour décider s'ils vont discuter avec vous ou non, et s'ils le font, quelle priorité ils vont vous donner
  - Pour un accès complet aux services sur Internet, la zone inversée est indispensable

Notre propre domaine

- Modifions named.conf.local :

```
zone "196.168.192.in-addr.arpa" {
 notify no;
 type master;
 file "pz/192.168.196";
};
```

Notre propre domaine

- C'est exactement comme pour le 0.16.172.in-addr.arpa et le contenu est similaire :

```
@     IN     SOA    ns.linux.bogus. hostmaster.linux.bogus. (
 199802151 ; Serial, todays date + todays serial
 8H     ; Refresh
 2H     ; Retry
 1W     ; Expire
 1D)    ; Minimum TTL
 NS     ns.linux.bogus.
1           PTR    gw.linux.bogus.
2           PTR    ns.linux.bogus.
3           PTR    donald.linux.bogus.
4           PTR    mail.linux.bogus.
5           PTR    ftp.linux.bogus.
```


Notre propre domaine

- Nous pouvons utiliser la commande « rndc » pour interagir avec notre serveur DNS (utilisez là sans argument pour connaître la liste des commandes disponibles) :

```
# rndc -b 172.16.0.1 -V status
```

-> doit retourner « server is up and running »

- Si cela ne fonctionne pas, vérifiez ce que donne la commande cat /var/run/bind/named.options
- Pour obtenir des statistiques détaillées sur votre DNS :

```
# rndc -b 172.16.0.1 stats
# cat /var/cache/bind/named.stats
```


Notre propre domaine

- Définissez au moins les machines www.m2i2l.org et ftp.m2i2l.org dans le fichier db.m2i2l.org de manière à effectuer les tests suivants :
  - la commande ifconfig doit mentionner eth0:0 utilisant l'adresse IP 172.16.0.1
  - la commande route doit mentionner le réseau m2i2l.org
  - vous pouvez aussi utiliser netstat -i ou netstat -nr
  - vous pouvez utiliser la commande host, nslookup ou dig :

```
# host www.m2i2l.org
    www.m2i2l.org is an alias for m2i2l.org.
    m2i2l.org has address 172.16.0.1
# dig ns1.m2i2l.org
    -> donne pleins d'infos et précise « AUTHORITY SECTION: m2i2l.org.
# nslookup ns1.m2i2l.org
    -> donne pleins d'info, voir le cours sur la partie relative à nslookup
```


Notre propre domaine

Testez depuis une machine différente de la machine sur laquelle vous avez installé votre DNS/DHCP/Apache2. Depuis cette machine distante, tapez les commandes suivantes (assurez-vous que la commande resolvconf soit installée) et remplacez eth0:1 par votre interface réseau :

```
# ifdown eth0:1
# ifconfig eth0:1 down
# ifconfig eth0:1 172.16.0.2 netmask 255.255.255.0 broadcast 172.16.0.255
up
# route add -net 172.16.0.0 netmask 255.255.255.0 gw 172.16.0.1 dev eth0:1
# route add default gw 172.16.0.1 netmask 255.255.255.0 metric 1 dev
eth0:1
# route
# ping 172.16.0.1
# resolvconf -a  m2i2l.org <<EOF
nameserver 172.16.0.1
search m2i2l.org
EOF
# host www.m2i2l.org
    www.m2i2l.org is an alias for m2i2l.org.
    m2i2l.org has address 172.16.0.1
```

Notre propre domaine

- Depuis la machine distante, ouvrez votre navigateur préféré et testez les url suivantes :
  - ftp://www.m2i2l.org
  - http://www.m2i2l.org
- Ca doit fonctionner...
- Réessayez le ftp après avoir tapé « **rndc -b 172.16.0.1 stop** » sur la machine et purgé votre navigateur ;
  Vérifiez que le port 53 n'est plus actif avec une commande « **map localhost** » sur la machine hébergeant le DNS ;
  Recommencez après avoir relancé via « **/etc/init.d/bind9 start** »
- Qu'est ce qui ne fonctionne pas ?
  -> Apache n'est toujours pas complètement configuré :
  -> Pas de VirtualHost
  -> Sécurité par défaut
  -> ...

DNS – Ouf c'est fini !

netfiltering et ipables

Préambule

- Netfilter / Iptables fonctionnent avec un kernel Linux 2.4 et supérieur
- Dans le cas d'un kernel Linux de type 2.2 ou inférieur, on doit utiliser une application appelée « **ipchains** »

Présentation

- Netfilter est le nom d'une partie du kernel Linux qui est destinée à assurer la surveillance de tous les transferts de données réseaux
- Sa tache est de faire du « Network Packet Filtering »
- Approximativement, il se place entre la couche réseau du kernel Linux, et la couche applicative

Présentation

- Netfilter supporte actuellement les protocoles IPv4, IPv6, DECnet, et ARP, et en partie IPX via des patchs expérimentaux
- Comme Netfilter est un élément implanté profondément dans le kernel Linux, on ne peut pas non plus le paramétrer directement via un fichier de configuration du « /etc/ »
- L'unique moyen à notre disposition est de dialoguer avec lui est le programme appelé « iptables »

Présentation

- On ne peut pas comparer le mode de fonctionnement des firewalls sous Windows et sous Linux tellement ils sont différents :
  - **Sous Windows,** c'est l'OS qui met à disposition les paquets IP qu'il désire dans l'espace utilisateur où un programme externe (« Tiny Firewall® », etc.) décide si oui ou non ce paquet doit être accepté, pas le système
  - **Sous Linux** par contre, tout reste au niveau de l'espace kernel, c'est-à-dire qu'à l'exception « d'iptables », aucun programme ne peut interférer avec Netfilter

Fonctionnement

- Netfilter peut intervenir en 5 endroits du système de gestion de la pile IP
- Pour chacune de ces étapes (que l'on appelle des « hook »), Netfilter peut :
  - Imposer au kernel de supprimer le paquet. Auquel cas, il est jeté aux oubliettes, et c'est comme si le paquet n'avait jamais existé
  - Indiquer au kernel que le paquet est accepté. Dans ce cas là, le kernel peut continuer à travailler dessus
  - Modifier le paquet, puis le rendre au kernel

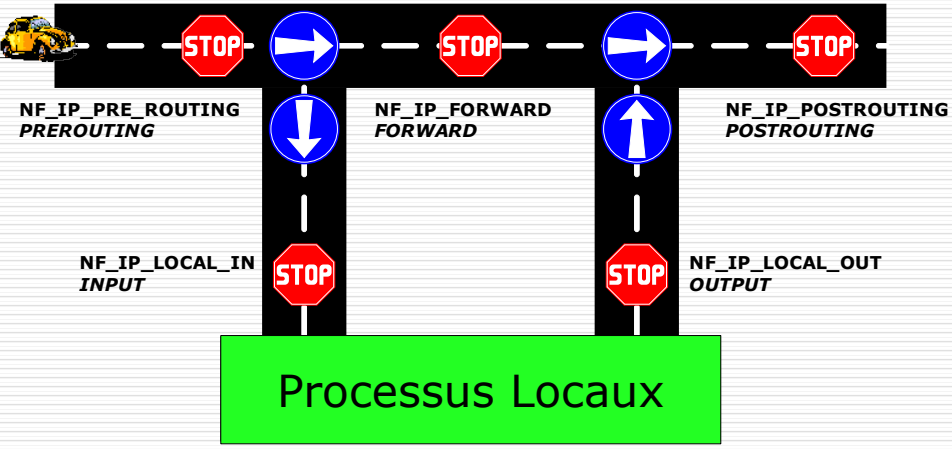
Fonctionnement

- Pour chacun de ces « hook », Netfilter associe une **chaîne**
- Une chaîne est un ensemble de règles (du type « si quelque chose alors je fais ceci ») concernant les paquets IP
  - Leur origine ;
  - Leur destination ;
  - Leur taille ;
  - Etc.
- En fonction des différentes règles de la chaîne, Netfilter pourra décider quoi fait du paquet IP :
  - Le laisser passer ;
  - Le supprimer ;
  - Le modifier.

Hook et chaines associés

Hook	Chaîne	Description
NF_IP_PRE_ROUTING	**PREROUTING**	A ce stade, le paquet est « brut de forme », c'est à dire qu'il n'a subit aucune modification par rapport à ce que l'interface réseau a reçu
NF_IP_LOCAL_IN	**INPUT**	A ce stade, le paquet est prêt à être envoyé aux couches applicatives, c'est à dire aux serveurs et aux clients qui tournent sur la machine
NF_IP_FORWARD	**FORWARD**	Ce « hook » voit passer des paquets IP qui vont transiter d'une interface réseau à une autre, **sans** passer par la couche applicative. Pourquoi faire suivre un paquet entre 2 interfaces réseaux, comme par exemple entre "eth0" et "ppp0" ? En fait, c'est afin de permettre à Linux de se transformer en passerelle
NF_IP_LOCAL_OUT	**OUPUT**	Ce « hook » est l'équivalent du « NF_IP_LOCAL_IN », sauf qu'il est exécuté après que les couches applicatives aient traités, ou générés, un paquet IP
NF_IP_POSTROUTING	**POSTROUTING**	C'est l'équivalent du « NF_IP_PRE_ROUTING » pour les paquets IP sortants de la couche IP. A ce stade, les paquets sont prêt à être envoyés sur l'interface réseau

Hook et chaines associés



Les tables

- Une table permet de définir un comportement précis de Netfilter
- Une table est en fait un ensemble de chaînes, elles-mêmes composées de règles
- Une table va nous permettre de manipuler Netfilter, afin de lui faire faire des choses intéressantes
- On manipule ces tables avec le programme « iptables »
- Il existe pour l'instant 4 tables :
  - Filter
  - NAT
  - Mangle
  - Raw -> Non étudiée « avec moi » en SRS/Réseaux
- D'autres pouvant être rajoutées à l'avenir

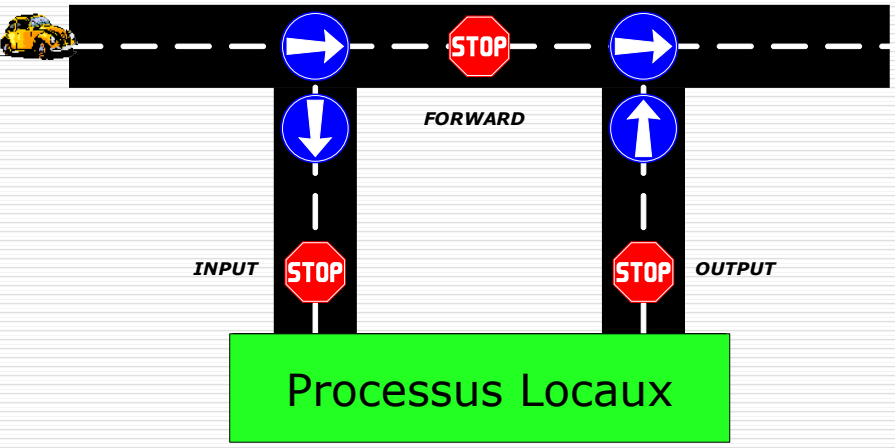
La table Filter

- Comme son nom l'indique, cette table sert à filtrer les paquets réseaux
- C'est à dire que nous pouvons trier les paquets qui passent à travers le réseau, et supprimer ceux qui ne nous intéressent pas, ou que nous trouvons dangereux
- Pour cela, la table « Filter » n'utilise que 3 chaînes :
  - **INPUT** : Cette chaîne contrôle les paquets à **destination** des applications (sockets locales)
  - **OUTPUT** : Elle analyse les paquets qui **sortent** des applications
  - **FORWARD** : Elle filtre les paquets qui passent d'une interface réseau à l'autre. Notez au passage que les paquets de ce type ne passent **jamais** par les chaînes INPUT et OUTPUT

La table Filter

- La philosophie du filtrage est très simple : Tout ce qui n'est pas **explicitement autorisé** est **strictement interdit**
- Pour cela, nous allons travailler en deux temps :
  - **Premièrement**, interdire par défaut tous les paquets. C'est facile à faire, car les 3 chaînes que nous utilisons (INPUT, OUTPUT et FORWARD) ont une valeur par défaut. Donc par défaut, nous allons supprimer toutes les trames (on utilisera par la suite le terme de « DROP »)
  - **Dans un second temps**, nous n'allons autoriser que certains flux bien particuliers. Ce sera un juste équilibre entre la sécurité du système et les fonctionnalités dont nous avons besoin

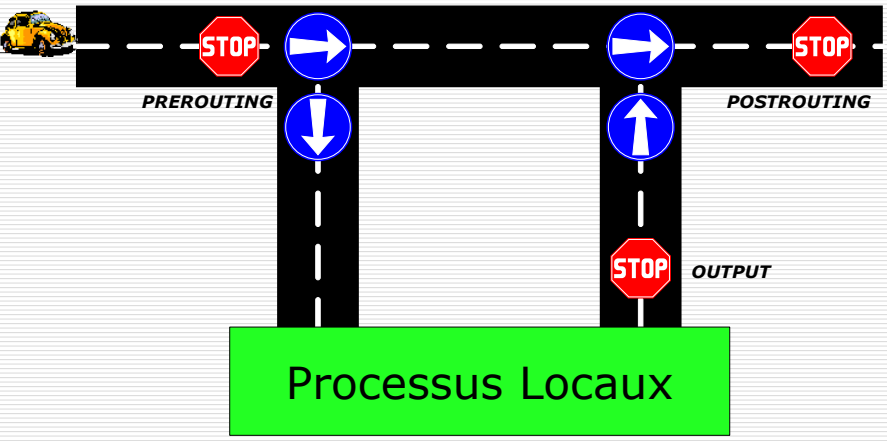
La table Filter



La table NAT

- La table NAT (**N**etwork **A**dress **T**ranslation) va transformer une machine Linux en une passerelle Internet, ce qui permettra à une autre machine de surfer sur Internet
- Pour faire tout ceci, on a besoin là encore de 3 chaînes :
  - **PREROUTING** : Les paquets vont être modifiés à l'entrée de la pile réseaux, et ce, qu'ils soient à destination des processus locaux où d'une autre interface.
  - **OUTPUT** : Les paquets sortant des processus locaux sont modifiés.
  - **POSTROUTING** : les paquets qui sont près à être envoyés aux interfaces réseaux sont modifiés.

La table NAT



La table Mangle

- le terme « mangle » en anglais veut dire « lacérer, mutiler, charcuter, déformer »
- il s'agit de les marquer en entrée de la couche réseau, afin que d'autres programmes de l'espace kernel (« kernel space ») puissent en faire quelque chose

La table Mangle

- L'idée de cette technique est par exemple de fournir à Linux la possibilité d'avoir un contrôle sur les débits des flux de données entrants et sortants de la machine, afin de rendre certains flux plus prioritaires que d'autres
- Certains flux vont être volontairement privilégiés, afin de garantir un meilleur confort d'utilisation à l'utilisateur

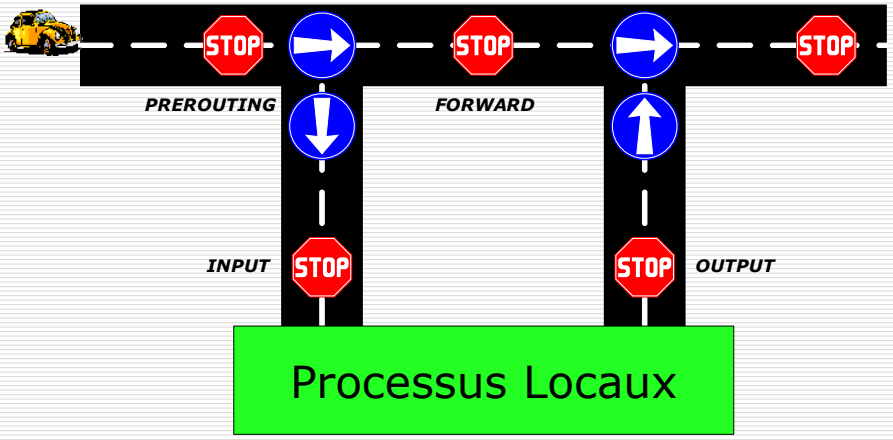
La table Mangle

- Exemple :
  - Tout en téléchargeant en FTP un gros fichier (comme une image ISO de la dernière distribution Linux à la mode), on peut désirer continuer à surfer tranquillement sur Internet, sans être tout le temps pénalisé par le gros téléchargement consommateur de bande passante
  - Ceci sera possible en modifiant la table Mangle et rendant le contenu HTTP (port source 80) prioritaire par rapport à celui de votre téléchargement (port source 20)

La table Mangle

- Dans les premiers kernels de la série 2.4, la table Mangle n'utilisait que 2 chaînes (PREROUTING et OUTPUT)
- Mais depuis le kernel 2.4.18 elle utilise toutes les chaînes de Netfilter :
  - **PREROUTING** : Les paquets vont être marqués en entrée de la couche réseau, en fonction de certains critères, de type de service (grâce aux numéros de ports source et/ou de destination), d'adresses IP de source et/ou de destination, de taille des paquets, etc. Ces informations seront utilisés par un programme fonctionnant dans l'espace kernel ;
  - **INPUT** : Les paquets sont marqués juste avant d'être envoyés aux processus locaux ;
  - **FORWARD** : Les paquets passant d'une interface réseau à l'autre sont marqués ;
  - **OUTPUT** : Là, ce sont les paquets générés par les applications locales (un client web par exemple) qui vont être marqués, tout comme les paquets entrant dans la couche réseau ;
  - **POSTROUTING** : Les paquets prêt à être envoyés sur le réseau sont marqués. L'utilisation de cette chaîne dans la table Mangle n'est cependant pas très évident.

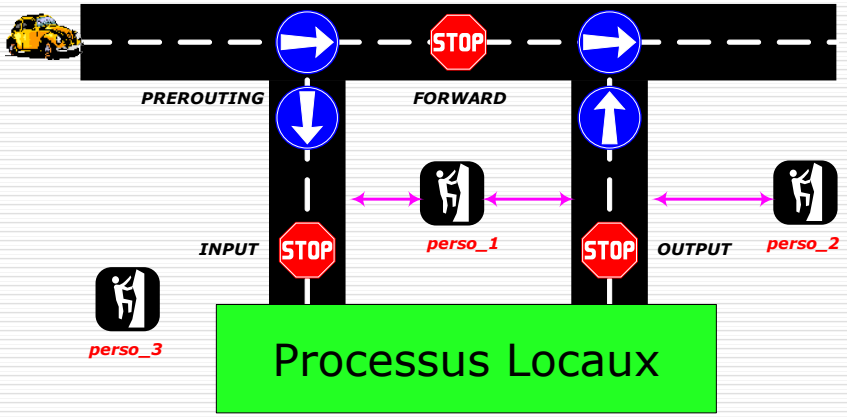
La table Mangle



Les chaines utilisateurs

- Il existe 5 principales chaînes, appelées aussi **chaînes pré-définies** (PREROUTING, INPUT, FORWARD, OUTPUT et POSTROUTING)
- Il est possible à l'utilisateur, à root pour être précis, de **créer ses propres chaînes**
- Ce sont les **chaînes utilisateurs**

Les chaines utilisateurs



Les chaines utilisateurs

- Sur le dessin précédent, on voit quelques chaînes utilisateurs qui sont créées pour la table « Filter »
- Comme on peut le constater, les chaînes utilisateurs peuvent :
  - Êtres utilisées par une chaîne en particulier : *perso_2*
  - Êtres appelées par plusieurs chaînes : *perso_1*
  - Ne pas êtres appelées du tout (*perso_3*). A quoi ça sert ? A rien… C'est pour l'exemple !

Règles et Cibles

- Les **règles**, comme leur nom l'indique, sont une série de critères auquel doivent ou non répondre les paquets
- Si le paquet réseau ressemble à l'un ou l'autre des critères, alors la règle est appliquée
- Les différentes règles d'une chaînes sont appliquées les unes à la suite des autres

Règles et Cibles

- Les **critères** peuvent être multiples :
  - Interface source ou destination
  - Adresse IP source ou de destination
  - Port source ou de destination
  - Type de trame
  - Nombre de paquets
  - Paquet marqué par la table Mangle
  - Etc

Règles et Cibles

- Enfin, à chaque règle est associée une **action** (ou « **CIBLE** » dans la nomenclature de Netfilter) à effectuer si la règle doit s'appliquer
- C'est là que Netfilter agit, qu'il fait quelque chose avec le paquet réseau

Règles et Cibles

- Les principales actions sont :
  - **DROP** : Le paquet est détruit purement et simplement
  - **ACCEPT** : Le paquet a une « bonne tête », il est donc autorisé à continuer à passer. Mais une autre règle située après la règle qui a accepté ce paquet peut très bien décider de le supprimer
  - **LOG / ULOG** : Le paquet est autorisé à continuer de passer, mais ses caractéristiques sont notées au passage. En général, c'est qu'on estime que le paquet est « louche », et que l'on veut en prendre note. Mais plus souvent encore, on décidera de le supprimer
  - **MASQUERADE** : Le paquet va être modifié, afin de dissimuler (de masquer en fait) certaines informations concernant son origine
  - **MARK** : le paquet est marqué en y attachant une information. Ceci est principalement utilisé avec les tables « Mangle »
  - **Une chaîne utilisateur** : Nous avons vu un peu plus haut qu'il existe des chaînes utilisateurs. Dans le cas de cette action, le paquet est envoyé à une chaîne définie par l'utilisateur, où il passera à travers de nouvelles règles

Iptables

- Iptables est donc une commande que seul root peut lancer
- Son but est de dialoguer avec Netfilter, afin de contrôler les **règles** des **chaînes**, dans le but de configurer les **tables**
- Iptables est la boîte à tout faire de Netfilter
- Cette commande va pouvoir :
  - **Rajouter** des règles / chaînes
  - **Supprimer** des règles / chaînes
  - **Modifier** des règles / chaînes
  - **Afficher** les règles / chaînes

Squid & SquidGuard

Définition

- Normalement, un navigateur web requérant une page web établit une liaison vers le serveur et rapatrie le contenu de la page spécifiée
- Cette opération s'effectue indépendamment de la distance entre le serveur et le client et donc sans considération pour la qualité globale de la transmission (à savoir la rapidité)

Définition

- En général, une page web est transmise plusieurs fois, lorsque plusieurs personnes totalement indépendantes, choisissent la même page
- Pour améliorer cela, les navigateurs web sont dotés d'un « cache » local

Définition

- Les pages web requises sont stockées dans le cache durant une période fixée par l'utilisateur
- Pour un réseau, un serveur proxy remplit la même tâche à l'aide d'un « cache local », et ce non seulement pour une machine individuelle, mais aussi pour l'ensemble des utilisateurs du réseau

Définition

- Toutes les pages web qui ont été rapatriées par les utilisateurs d'un même réseau sont immédiatement à disposition, car elles n'ont pas besoin d'être retransmises depuis Internet

Pourquoi ?

- **Pourquoi utiliser un cache ?**
  - Pour soulager le réseau
  - Ouvrir l'internet aux machines situées derrière une pare-feu (firewall)
  - Contrôler l'utilisation
  - Restreindre les accès

Pourquoi ?

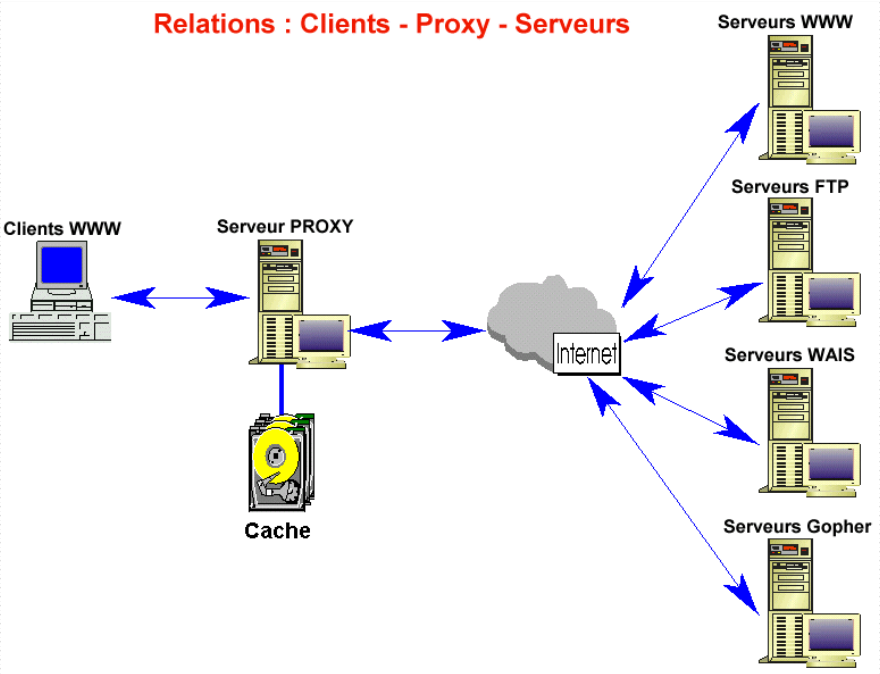
- **Validité du contenu d'un cache WWW :** les documents stockés dans un cache y demeurent en fonction :
  - D'une date d'expiration (en-tête HTTP expire)
  - D'une durée de vie estimée à partir des dates de dernière modification (en-tête HTTP Last-Modified)
  - D'une durée de vie fixée arbitrairement (HTTP/0.9, FTP)
  - de la fréquence d'utilisation du document
  - Des documents ne doivent jamais être cachés (les procédures CGI, les documents soumis à identification d'utilisateur, …)

Son rôle

- Faire du cache
- Filtrer certains sites : On peut interdire certaines adresses sur internet avec les proxy actuels
- Il existe pour certains proxy des bases de sites pornographiques, des chats, …
- Interdire l'accès d'internet à certaines de vos machines :
  - On peut interdire l'accès d'internet à certaines machines sans pour autant interdire l'accès à notre intranet

Son rôle

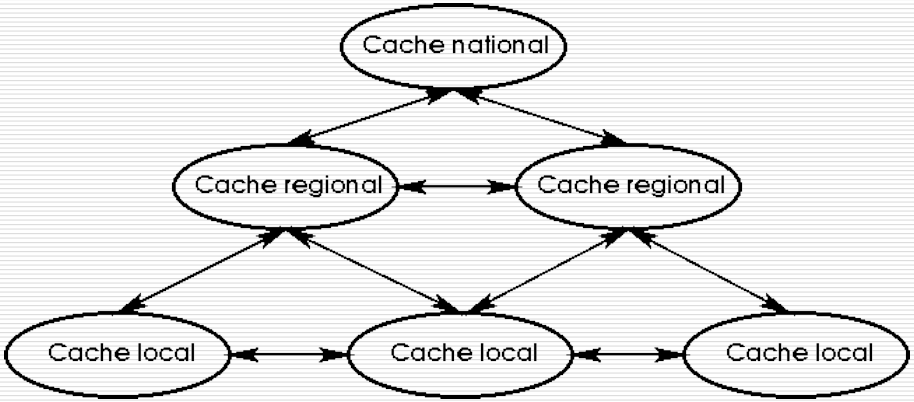
- Interdire le téléchargement : si les étudiants essayent de télécharger des programmes, jeux .... On peut l'interdire avec le proxy
- Protéger notre réseau : Cela nous permet de sécuriser nos machines qui se trouvent derrière par rapport à l'internet
  - En effet le proxy permet d'interdire certains ports particulièrement sensibles quant à la sécurité
- Partager l'accès à internet



Réseau de caches

- Les serveurs caches tel que SQUID ont la possibilité de s'associer en réseau et forme ainsi une hiérarchie (exemple de Renater-Cache)
- On distingue deux types de liaisons inter-caches :
  - Hiérarchiques, des mises à jour de cache s'effectuent en cascade.
  - Transversales, simples sondages des caches voisins.
- Les liaisons de voisinage donnent des résultats (taux de succès dans les accès) très aléatoires (à conseiller en cas de d'homogénéité de communauté, réseau de caches thématiques...)
- La construction d'un réseau de caches WWW hiérarchique s'opère en respectant les contraintes suivantes :
  - A chaque niveau de la hiérarchie, les caches doivent être installés sur les tronçons les plus rapides du réseau utilisé
  - Le dimensionnement de la taille d'un cache de niveau n doit tenir compte de la taille des caches de niveau n-1 ainsi que du nombre d'accès journaliers de chacun d'entre eux.

Réseau de caches



Présentation

- SQUID est un proxy WEB puissant avec de très nombreuses options
- SquidGuard est un plugin qui permet de rediriger certaines pages vers des pages choisies, afin d'interdire la consultation de certaines pages

Squid

- Squid assure les fonctions suivantes :
  - Un cache WEB commun pour optimiser la bande passante
  - La possibilité d'introduire une authentification utilisateurs afin d'interdire le surf à certains utilisateurs
  - Un filtrage d'accès « basique »

SquidGuard

- SquidGuard propose un filtrage puissant d'accès au web, en fonction :
  - De groupes d'utilisateurs, définis de diverses manières. Par exemple, on peut se baser sur des IPs ou des groupes d'IPs, mais il est possible d'utiliser l'authentification des utilisateurs mise en place sur Squid
  - De listes de domaines et d'URI (Uniform Resource Identifier) qui serviront à définir soit des cibles autorisées, soit des cibles interdites
  - De listes de domaines et d'URI qui ne serviront qu'à interdire l'accès aux cibles spécifiées
  - De plages horaires pendant lesquelles l'accès sera autorisé ou interdit

Installation

- Tout simplement :
  aptitude install squid squidguard
- si vous désirez un outil graphique/GTK pour configurer squid, vous devez installer le package gadmin-squid
  (voir également gadmin-squid, squidtaild, squidview, srg, sarg...)
- Squid est un daemon, il se comporte donc comme n'importe quel daemon
  - Pour démarrer : */etc/init.d/squid start*
  - Pour l'arrêter : */etc/init.d/squid stop*
  - Pour le forcer à relire sa config. : */etc/init.d/squid reload*
    (après modification de la configuration par exemple)
  - Pour le redémarrer : */etc/init.d/squid restart*
- Les fichiers de logs sont par défaut dans : */var/log/squid*
  n'oubliez pas de les consulter en cas de problèmes

La configuration de Squid

- Le fichier de configuration est : */etc/squid/squid.conf*
- Nous n'allons pas tout lister, mais seulement quelques options les plus générales

squid.conf

hierarchy_stoplist cgi-bin ?

- Le mot clef « hierarchy_stoplist » définit une liste de mots qui, si elle trouvée dans l'url, sera prise en charge directement par le cache

```
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
```

Le résultat de ces lignes de configuration est que les pages résultant de script CGI ne sont pas mises en cache.

- Une liste d'ACL qui si elle est détectée fera que la requête ne sera pas acceptée
- Si on utilise DENY, le nom de l'ACL ne sera pas mis dans le cache

squid.conf

cache_mem 21 MB

- Taille maximum de mémoire vive utilisée pour stocker du cache

maximum_object_size 20 MB

- Taille maximum des objets stockés dans le cache

squid.conf

cache_dir ufs /var/spool/squid 100 16 256

- Chemin des fichiers de cache
- Ici on définit les répertoires du cache, on peut définir plusieurs répertoires, l'intérêt est de définir des répertoires sur des disques différents afin de gagner en vitesse

squid.conf

emulate_httpd_log off

- Format des logs :
  - Avec off, squid utilise son propre format de logs, mais la date et l'heure ne sont pas lisible
  - Avec on, squid utilise le format standard CLF

squid.conf

redirect_program /usr/bin/squidGuard
redirect_children 4

- Ces deux lignes permettent d'intégrer le plugin SquidGuard

squid.conf

```
acl ReseauI2L src 172.16.0.0/255.255.255.0
```

- ACL qui définit le réseau utilisant le cache
- Ici on définit les ACL, ceci vous permet de définir des groupes afin de leurs attribuer des droits différents
- Cette ligne définit les adresses IP des utilisateurs du réseau
- On doit la personnaliser selon sa configuration réseau

squid.conf

```
# acl all src 0.0.0.0/0.0.0.0
acl all src all
acl manager proto cache_object
# acl localhost src 127.0.0.1/255.255.255.255
acl localhost src 127.0.0.1/32
acl to_localhost dst 127.0.0.0/8 0.0.0.0/32
acl SSL_ports port 443 563        # https, snews
```

- Liste des ACL par défaut → A conserver

squid.conf

```
#acl SSL_ports port 873          # rsync
acl SSL_ports port 443           # https
#acl SSL_ports port 563          # snews
acl Safe_ports port 80           # http
acl Safe_ports port 21           # ftp
acl Safe_ports port 443          # https
#acl Safe_ports port 22          # ssh
#acl Safe_ports port 1863        # msn
#acl Safe_ports port 70          # gopher
#acl Safe_ports port 210         # wais
#acl Safe_ports port 1025-65535  # unregistered ports
#acl Safe_ports port 280         # http-mgmt
#acl Safe_ports port 488         # gss-http
#acl Safe_ports port 591         # filemaker
#acl Safe_ports port 777         # multiling http
#acl Safe_ports port 631         # cups
#acl Safe_ports port 873         # rsync
#acl Safe_ports port 901         # SWAT
```

- Cette partie définir les ports auquels certains utilisateurs auront droits d'accès, ici ils auront accès à http (port 80), https (port 443), ftp (21)...
- On peut, bien entendu, interdire les accès en commentant ou supprimant les lignes correspondantes

squid.conf

- Enfin nous définissons les accès
- Les règles sont lues dans l'ordre en s'arrêtant à la première interdiction correspondant au critère de l'utilisateur, un peu comme un firewall, l'ordre des règles est donc important !
- Dans l'exemple, ci dessous, localhost a tous les droits
- La ligne ***http_access deny !Safe_ports*** interdit les accès à tous les ports exceptés ceux définis par les lignes *acl Safe_ports*
- *La ligne* ***http_access deny CONNECT !SSL_ports*** *interdit les connexions à tous les ports exceptés ceux définis par les lignes acl SSL_ports*

```
acl purge method PURGE
acl CONNECT method CONNECT

# Accès fournis par squid
http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
```

Manager autorisé sur localhost
Non autorisé sinon

Purge autorisée sur localhost
Non autorisée sinon

squid.conf

- On donne ensuite accès au WEB au localhost et au réseau « Mon réseau » définit précédemment

```
http_access allow ReseauI2L
http_access allow localhost
```

- **On interdit le reste, important car sinon notre proxy sera un « open proxy »**
  i.e. il pourra être utilisé par n'importe qui via internet, notamment pour faire des choses pas biens

```
http_access deny all
```


squid.conf

- Dans les précédentes version de squid, ces lignes sont importantes dans le cas d'un proxy transparent ne les oublier pas (on va y revenir un peu plus tard)

```
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
httpd_accel_host virtual
httpd_accel_port 80
```

- Avec les versions récentes de squid, il suffit de préciser :

```
http_port 3128 transparent
```


squid.conf

- On doit indiquer les serveurs DNS
  -> Selon ce que vous voulez faire avec le proxy (selon le(s) réseau(x) pris en charge)
  -> Il faut préciser les DNS vus lors du TP1 SRS/Kernel :

```
dns_nameservers 195.220.130.2 192.220.130.10
```

  -> Ou « votre » DNS :

```
dns_nameservers 172.16.0.1
```

  Si on désire effectuer des statistiques WEB notamment avec MRTG ou rrdtool, Squid peut faire serveur SNMP

```
acl snmppublic snmp_community public
snmp_port 3401
#snmp_access allow snmppublic all -> dangereux !
snmp_access allow snmppublic localhost
snmp_access deny all
```


squid.conf

- On définit ensuite l'host de notre proxy et le mail de l'administrateur

```
visible_hostname monproxy.domain
cache_mgr admin@domain
```

Rendre le proxy transparent – Pourquoi ?

- Par les temps qui courent, il n'est pas rare que les connections internet soient saturées par les téléchargements, lorsque plusieurs machines utilisent la même connexion internet
- Depuis longtemps déjà, on utilise des serveurs proxy (en général HTTP)
- L'utilisateur d'un réseau d'entreprise devait entrer dans son navigateur l'adresse du proxy pour espérer accéder à internet

Rendre le proxy transparent – Pourquoi ?

- On entend parler également de proxy universel, qui désigne les serveurs proxy de type SOCKS
- Ce qui permet d'étendre les capacités d'un serveur proxy à un plus grand nombre d'applications

Rendre le proxy transparent – Pourquoi ?

- Une autre solution pour partager une connexion internet consiste a utiliser un routeur pratiquant ce que l'on appelle du NAT
- Ceci ne nécessite pas de configuration supplémentaire, lorsque le réseau est configuré, mais permet à quasiment toute application de se connecter à internet

Rendre le proxy transparent – Pourquoi ?

- le proxy transparent apparait être un bon compromis entre configuration et filtrage des applications
- Celui ci agit sur le réseau comme s'il s'agissait d'un routeur
  - Pas de configuration superflue de la part de l'utilisateur
- Mais filtre comme un serveur proxy
  - bloque les applications autres que celles définies par l'administrateur

Rendre le proxy transparent – Pourquoi ?

- Dans le cadre d'une entreprise, un proxy transparent est utile pour les raisons suivantes :
  - Mettre en cache les pages les plus visitées, afin d'éviter de recharger une page qui a déjà été visitée
  - Forcer les utilisateurs du réseau à utiliser un proxy, qu'ils le veuillent ou non
  - Faire utiliser un proxy a toutes les machines d'un réseau sans avoir a configurer chaque application
  - Bloquer des applications internet (comme les utilitaires de chat, ou de peer2peer)
  - Sécuriser un réseau qui accède a internet en limitant l'accès a certaines pages.

Rendre le proxy transparent – Pourquoi ?

- Le principe est relativement simple
  - Le réseau est configuré avec une passerelle par défaut et un serveur DNS
  - Lorsqu'une machine tente d'accéder à une page internet elle envoie une requête sur un serveur sur le port 80
  - Cette requête est automatiquement, et de manière invisible interceptée par le proxy
  - Le serveur proxy relaye la requête sur le serveur de la page internet et stocke la page sur son disque dur
  - La machine du visiteur croit donc dialoguer avec le serveur internet, mais en réalité, elle ne dialogue qu'avec le serveur proxy
  - Si une autre machine a déjà demandé la même page internet, le proxy va pouvoir lui fournir depuis son disque dur (après avoir tout de même vérifié que la page internet n'a pas subi de modification), évitant ainsi toute attente, et sans que l'utilisateur ne s'aperçoive de quelque chose

Rendre le proxy transparent - Première solution

- Squid et iptables sur la même machine
- Cette première méthode est la plus simple à mettre en œuvre
  - Elle tient en une ligne, par contre , elle est moins sécurisée car le serveur proxy est accessible directement via le port TCP 3128
  - Un petit malin peut parfaitement utiliser un tunnel http pour utiliser ses applications favorites …

Rendre le proxy transparent - Première solution

- Pour activer le proxy transparent il suffit de taper la ligne suivante

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

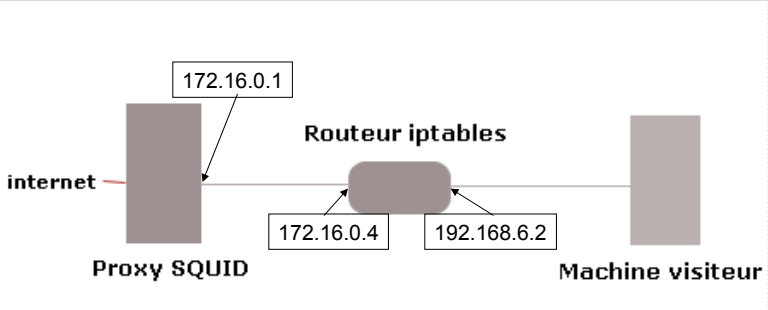
- Cette ligne indique à la machine d'intercepter toutes les requêtes sur un port 80, et de les rediriger vers le proxy
- Avec les deux lignes citées précédemment , les machines clientes de notre réseau local qui tentent d'accéder par exemple à www.google.fr sur le port 80 seront tout simplement redirigées sur le serveur proxy http

Rendre le proxy transparent - Première solution

- Mais par contre, des programmes comme ping, telnet, ou autres ssh ne fonctionnent pas, et donc par conséquent, toutes les applications de type peer2peer sont également bloquées

Rendre le proxy transparent - Deuxième solution

- Routeur et Squid sur deux machines différentes



Rendre le proxy transparent - Deuxième solution

- Cette méthode permet d'augmenter la sécurité car le proxy n'est plus accessible directement par les machines clientes
- Imaginons donc que le routeur iptables possède l'adresse IP 192.168.6.2 du coté des machines clientes, et l'adresse IP 172.16.0.4 du coté proxy
- Et que le proxy possède l'adresse IP 172.16.0.1
- Nous devrons configurer la table iptables du routeur de la facon suivante :

Rendre le proxy transparent - Deuxième solution

- Nous devrons configurer la table iptables du routeur de la façon suivante :

```
iptables -t nat -A PREROUTING -i eth0 -s ! 172.16.0.1 -p tcp --dport 80 -j DNAT --to 172.16.0.1:3128
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.6.0/24 -d 172.16.0.1 -j SNAT --to 172.16.0.4
iptables -A FORWARD -s 192.168.6.0/24 -d 172.16.0.1 -i eth0 -o eth1 -p tcp --dport 3128 -j ACCEPT
```

- ce qui signifie que les paquets ne provenant pas du proxy mais étant a destination du port 80 seront redirigés vers le proxy ; que les paquets revenant depuis le proxy, passeront obligatoirement par le routeur ; et que nous autorisons les paquets provenant du réseau local à être redirigés vers le proxy

Rendre le proxy transparent - Deuxième solution

- Seulement dans ces deux dernières configurations, des problèmes peuvent se poser si le proxy est directement connecté à internet (utilisation du proxy par d'autres internautes externes au réseau), auquel cas nous allons avoir recours aux Access Control Lists (ACL) de Squid

Rendre le proxy transparent - Deuxième solution

- Dans le dernier cas cité, nous opérons de la manière suivante : en éditant le fichier squid.conf

```
acl machines_locales src 172.16.0.0/16
http_access allow machines_locales
http_access deny !machines_locales
```

- Et dans le premier cas :

```
acl machines_locales src 192.168.6.0/24
http_access allow machines_locales
http_access deny !machines_locales
```

-> Vérifiez ces acl ne sont pas déjà dans squid.conf !

-> Adaptez ces ACL à votre configuration !

La configuration de SquidGuard

- Le fichier de configuration est : */etc/squid/squidGuard.conf*
- Nous n'allons pas tout lister, mais seulement quelques options les plus générales

squidGuard.conf

- Ici on définit les répertoires où se trouvent les blacklist et le répertoire des logs

```
dbhome /var/lib/squidguard/db
logdir /var/log/squid
```

squidGuard.conf

- On défini quand on peut travailler

```
# TIME RULES:
# abbrev for weekdays:
# s = sun, m = mon, t =tue, w = wed, h = thu, f =
fri, a = sat

time workhours {
        weekly s 09:30-12:00 13:00-19:00
        weekly m 09:00-12:00 13:00-19:00
        weekly t 09:00-11:00 12:00-19:00
        weekly w 09:00-12:00 12:00-18:00
        weekly h 09:00-13:00 13:00-18:00
        weekly f 09:00-12:00 13:30-18:00
        weekly a 08:20-13:00 13:30-19:00
}
```


squidGuard.conf

- On définit en suite les groupes de listes contenants
  - Les domains (domainlist) soit donc les domaines interdits
  - La liste des urls (urllist) soit donc les URL interdites
  - La liste des expression (expressionlist) soit donc les expressions interdites contenues dans une page WEB
  - L'option *log /var/log/squid/nom_du_fichier.log* permet de générer un log lorsqu'une redirection est effectuée

squidGuard.conf

```
dest redirector {
    domainlist redirector/domains
    urllist redirector/urls
    expressionlist  redirector/expressions
}

dest warez {
    domainlist warez/domains
   urllist warez/urls
}

dest ads {
    domainlist ads/domains
    urllist ads/urls
}

dest aggressive {
    domainlist aggressive/domains
    urllist aggressive/urls
}
```

```
dest drugs {
    domainlist drugs/domains
    urllist drugs/urls
}

dest gambling {
    domainlist gambling/domains
    urllist gambling/urls
    log /var/log/squid/jeux.log
}

dest violence {
    domainlist violence/domains
    urllist violence/urls
    expressionlist  violence/expressions
}
```

squidGuard.conf

- On peut ensuite définir des groupes

```
source LAN {
ip 10.1.2.0/255.255.2550
}
```

- On définit ensuite les droits pour chaque groupes et la page vers laquelle ils seront redirigés
- L'option pass !groupe_de_listes permet de donner l'accès si cela ne correspond pas aux listes interdites et l'adresse de la page vers laquelle ils sont redirigés

squidGuard.conf

- On doit donc avoir un serveur HTTP installé, par défaut, la page fournie par SquidGuard (/usr/share/doc/squidguard/examples /squidGuard.cgi.gz) est en cgi.
- On doit donc la copier dans le dossier cgi (par défaut sur apache /usr/lib/cgi-bin/) et activer le cgi sur notre serveur HTTP

squidGuard.conf

```
# ACLs
acl {
      LAN {
            pass !adult !audio-video !hacking !redirector !warez !ads !
aggressive !drugs !gambling !violence all
            redirect http://127.0.0.1/cgi-bin/squidGuard.cgi?clientaddr=
%a&srcclass=%s&targetclass=%t&url=%u&lang=fr
      }
      Default
      redirect http://127.0.0.1/cgi-bin/squidGuard.cgi?clientaddr=
%a&srcclass=%s&targetclass=%t&url=%u
    pass none
    }
}
```


Squid et ses logs

- On peut surveiller les logs à l'aide de la commande « tail -f access.log »
- les logs de Squid se trouvant dans le répertoire /var/log/squid
  - « access.log » donne les informations sur les requêtes qui ont transité par Squid
  - « cache.log » informe sur l'état du serveur lors de son démarrage
  - « store.log » informe sur les objets stockés dans le cache.

Squid et ses logs

- Voici les valeurs que vous pouvez voir dans le fichier access.log
- Les codes commençant par TCP_ sont les requêtes sur le port 8080

TCP_HIT	Une copie valide se trouve dans le cache
TCP_MISS	Pas dans le cache
TCP_REFRESH_HIT	Objet dans le cache, mais périmé. Squid demande au serveur d'origine si une nouvelle version est disponible, la réponse étant pas de nouvelle version
TCP_REF_FAIL_HIT	Objet dans le cache, mais périmé. Squid demande une mise à jour, mais n'obtient pas de réponse du serveur. Il renvoie alors l'ancienne version
TCP_REFRESH_MISS	Objet dans le cache, mais périmé. Squid demande une mise à jour qu'il reçoit
TCP_CLIENT_REFRESH	Le client envoie une requête avec une demande de ne pas utiliser le cache. Squid forwarde la requête
TCP_IMS_HIT	Le client demande une mise à jour, et l'objet est dans le cache et est récent. Squid ne forwarde pas la requête
TCP_IMS_MISS	Le client demande une mise à jour. Squid forwarde la requête
TCP_SWAPFAIL	Problème de swap. L'objet semble être dans le cache mais n'est pas accessible. La requête est forwardée
TCP_DENIED	Accès est interdit

Squid et ses logs

- UDP_ sont des codes sur le port ICP

UDP_HIT	Une copie récente de la copie est dans le cache
UDP_HIT_OBJET	Idem que UDP_HIT, mais l'objet est envoyé dans un paquet UDP
UDP_MISS	Objet pas dans le cache ou périmé
UDP_DENIED	Accès interdit pour cette requête
UDP_INVALID	Requête invalide
UDP_MISS_NOFETCH	La requête n'a pas était faite à temps (arrêt ou démarrage du serveur)

Contrôler le cache

- Réinitialiser le cache, pour cela il faut relancer Squid avec l'option –z
- Purger le cache, pour cela il faut mettre dans le fichier « squid.conf » les ACL suivantes afin de n'autoriser cela que depuis la machine serveur

```
acl PURGE method purge
acl localhost src 127.0.0.1
http_access allow purge localhost
http_access deny purge
```


Port Knocking

Question de sécurité

- Un système totalement sécurisé vis-à-vis du réseau est un système qui n'y est pas connecté
- Malheureusement, ce n'est pas très intéressant
- Il faut donc le connecter et lui appliquer plusieurs couches de sécurité afin de le rendre le moins vulnérable possible

Question de sécurité

- Le protocole **Port Knocking** propose une couche de sécurité supplémentaire
  - Un firewall permet d'authentifier un hôte sur le réseau
  - Le protocole **PortKnocking** permet d'authentifier un utilisateur de ce réseau

Question de sécurité

- Notez que le protocole **Port Knocking** ne connait rien au filtrage proprement dit, il ne fait qu'exécuter des commandes externes
- Il est donc indépendant d'un logiciel de filtrage utilisé, voire même de l'exact usage qui en sera fait → Ce n'est pas un firewall !
- On peut aussi bien le faire tourner sur la machine ayant le service SSH que sur un garde-barrière situé en amont

Une forme d'encodage

- Le protocole **Port Knocking** décrit une communication au sein de laquelle les informations arrivent encodées sous forme de tentatives de connexion à des ports fermés
- Cette séquence de ports forme un code, et déclenche un évènement sur le récepteur
- Le récepteur ne retourne rien à l'émetteur durant cette transaction

Une forme d'encodage

- Le protocole **Port Knocking** est une méthode de communication entre deux ordinateurs
- L'information est codée dans une séquence de tentatives de connexion
- Cette séquence se nomme « knocks »
- Au commencement, le serveur ne présente aucun port ouvert et surveille toutes les tentatives de connexion

Une forme d'encodage

- Le client lance des tentatives de connexion vers le serveur en envoyant des paquets SYN aux ports indiqués dans les « knocks »
- Ce processus de « frappement » est ce qui donne au protocole **Port Knocking** son nom

Une forme d'encodage

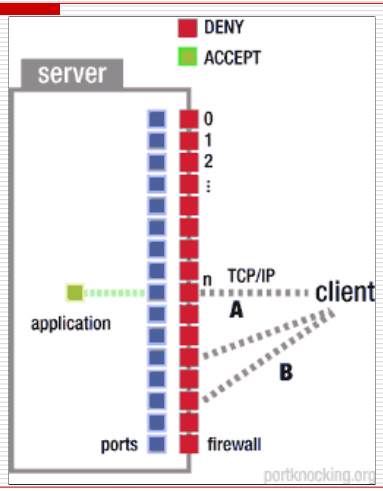
- Le serveur n'offre aucune réponse au client pendant cette phase, car il traite « silencieusement » la séquence
- Quand le serveur décode un knock valide, il déclenche un processus défini

Une forme d'encodage

- La définition des « knocks » valides est arbitraire, et définie par l'administrateur
- Le processus déclenché sur le serveur est également arbitraire, et défini par l'administrateur
- Le déclenchement peut avoir comme conséquence la modification dynamique des règles de filtrage ou d'autres commandes du système

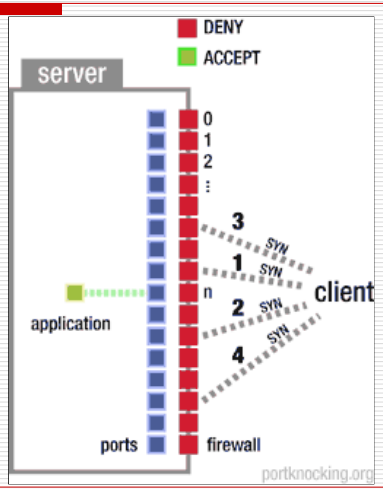
Fonctionnement

- (A) Le client ne peut pas se connecter à l'application qui écoute sur le port « n »
- (B) le client ne peut établir de connexion sur quelque port que ce soit



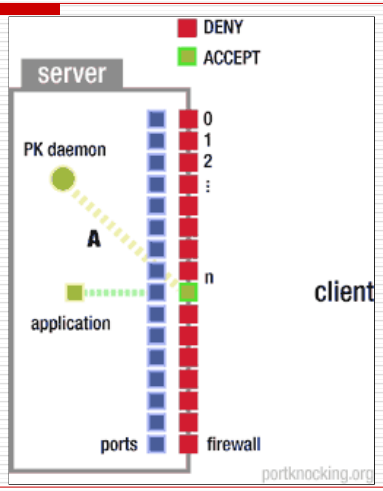
Fonctionnement

- (1,2,3,4) Le client se connecte (eg. envoie des paquets SYN) à un ensemble de ports pré-définis dans une séquence qui représente un message crypté
- Le client doit connaitre au préalable l'existence du protocole **Port Knocking** et de sa configuration, mais il ne reçoit aucun retour d'information durant cette phase car les règles de filtrage du pare-feu l'en empêche



Fonctionnement

- (A) Le processus système intercepte les tentatives de connexion et les interprète (éventuellement, il les décrypte et les décode) et détermine que ce comportement correspond à sa configuration
- Le serveur exécute alors une tâche spécifique grâce au contenu du message, telle qu'ouvrir le port n pour ce client



Fonctionnement

- (A) Le client se connecte au port n et communique avec l'application sous-jacente par son mécanisme habituel