

## Aspects Réseaux sous Linux

# COMPLEMENTS ET RAPPELS

(CentOs)



Jean-Louis Gouwy



# Compléments et rappels

## Plan

- Les modes de mise en réseau des interfaces sous VirtualBox
- Les outils d'audit
- Fichiers particuliers
- Fichiers de configuration
- Les expressions régulières



# Compléments et rappels

## Les modes de mise en réseau des interfaces sous VirtualBox

- **Réseau privé hôte**

Les VM peuvent seulement communiquer ensemble et avec l'hôte uniquement.

- **Nat**

Les VM peuvent communiquer ensemble, avec l'hôte, les machines du réseau physique et l'extérieur.

- **Réseaux Nat**

Les VM peuvent communiquer au sein d'un même réseau NAT, avec l'hôte, les machines du réseau physique et l'extérieur.

- **Bridge (Pont)**

Les VM sont disposées sur le même réseau physique que l'hôte.

- **Réseau interne**

Les VM sont reliées entre elles mais isolées de l'extérieur.



# Compléments et rappels

## Les outils d'audit

- **Liste des paquets contenant les commandes étudiées ci-après**

<b>Commande</b>	<b>Paquet</b>
dmesg	util-linux
ifconfig (*)	net-tools
route (*)	net-tools
netstat	net-tools
arp	net-tools
ping	net-tools
traceroute	traceroute
ip	iproute
ss	iproute

(\*) Considérée comme obsolète mais encore très utilisé par les sysadmin ...



# Compléments et rappels

## Les outils d'audit

- **dmesg**

Voir ce que le kernel a reconnu comme matériel.

```
# dmesg | grep eth0
```

- **ifconfig ou ip**

Donne la liste de toutes les interfaces et leur configuration.

```
# ifconfig  
# ifconfig -a  
# ip a s
```

Montage et démontage d'une interface à la volée.

```
# ifconfig eth0 down  
# ip link set dev eth0 down  
# ifconfig eth0 up  
# ip link set dev eth0 up
```



# Compléments et rappels

## Les outils d'audit

- **ifconfig** ou **ip**

Modification des paramètres Ip d'une interface à la volée.

```
# ifconfig eth0 192.168.0.3 netmask 255.255.255.0
```

```
# ip addr add 192.168.0.3/24 broadcast 192.168.0.255 dev eth0
# ip link set dev eth0 up
```

Changement de l'adresse Mac de l'interface.

```
# ifconfig eth0 hw ether 12:34:56:78:90:ab
# ip link set eth0 address 12:34:56:78:90:ab
```

Changement de la valeur du MTU

```
# ifconfig eth0 mtu 1500
# ip link set dev eth0 mtu 1500
```



# Compléments et rappels

## Les outils d'audit

- **route** ou **ip** ou **netstat**

Voir les routes actives dans le système.

```
# route  
# route -n  
# netstat -r  
# netstat -nr  
# ip r s
```

Ajout / suppression de routes à la volée.

```
# route add -net 192.168.8.0 netmask 255.255.255.0 gw 10.1.0.1 eth0  
# route del -net 192.168.8.0 netmask 255.255.255.0 gw 10.1.0.1 eth0  
# route add default gw 10.1.0.1 eth0  
# route del default  
  
# route add -host 192.168.7.5 gw 10.1.0.1 eth0  
  
...
```



# Compléments et rappels

## Les outils d'audit

- **route** ou **ip** ou **netstat**

Ajout / suppression de routes à la volée.

```
# ip r add 192.168.8.0/24 via 10.1.0.1 dev eth0
# ip r del 192.168.8.0/24

# ip r add default via 10.1.0.1 dev eth0 proto static
# ip r del default

# ip r add 192.168.7.5/32 via 10.1.0.1 dev eth0
...
```



# Compléments et rappels

## Les outils d'audit

- **netstat** (ou **ss**)

Outil intégrant plusieurs fonctions.

```
# netstat -l  
# netstat -nl  
  
# netstat -tul  
# netstat -tunl  
  
# netstat -tupan  
# netstat -tu  
  
...
```



# Compléments et rappels

## Les outils d'audit

- **arp** ou **ip**

Pour voir et manipuler le cache arp.

```
# arp -a
# ip neigh

# arp -s 192.168.5.5 7f:8b:56:83:90:ab
# ip neigh add 192.168.5.5 lladdr 7f:8b:56:83:90:ab nud
                                         permanent dev eth0

...
```



# Compléments et rappels

## Les outils d'audit

- **ping**

Envoi et réception d'un paquet icmp.

```
# ping 192.168.5.5
# ping -c 2 192.168.5.5
# ping -t 2 192.168.5.5
# ping -s 504 192.168.5.5
# ping 192.168.5.5 -M do -s 1472
```

- **traceroute**

Suivre le cheminement d'un paquet sur le réseau.

```
# traceroute www.helha.be
```



# Compléments et rappels

## Fichiers particuliers

- **/proc/net/route** Envoi et réception d'un paquet icmp.
- **/proc/net/arp** Cache arp.
- **/proc/sys/net/ipv4/ip\_forward** (Dés)activation del'ip forwarding.
- **/proc/sys/net/ipv4/icmp\_echo\_ignore\_all**  
Acceptation/refus d'un echo request.
- **/proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcast**  
Acceptation/refus d'un echo request diffusé.
- **/etc/sysctl.conf** (Dés)activation del'ip forwarding.
- **/etc/hosts** Résolution de noms de machines.
- **/etc/networks** Résolution de noms de réseaux.
- **/etc/services** Association n°/nom de service.
- **/etc/protocols** Association n°/nom de protocole.



# Compléments et rappels

## Fichiers de configuration

- **/etc/sysconfig/network**

Montage du réseau + Nom de la machine + Route par défaut

- **/etc/sysconfig/network-scripts/ifcfg-eth<sub>x</sub>**

Configuration d'une interface ethernet :

ip, masque, montage automatique, type d'attribution de l'ip ...

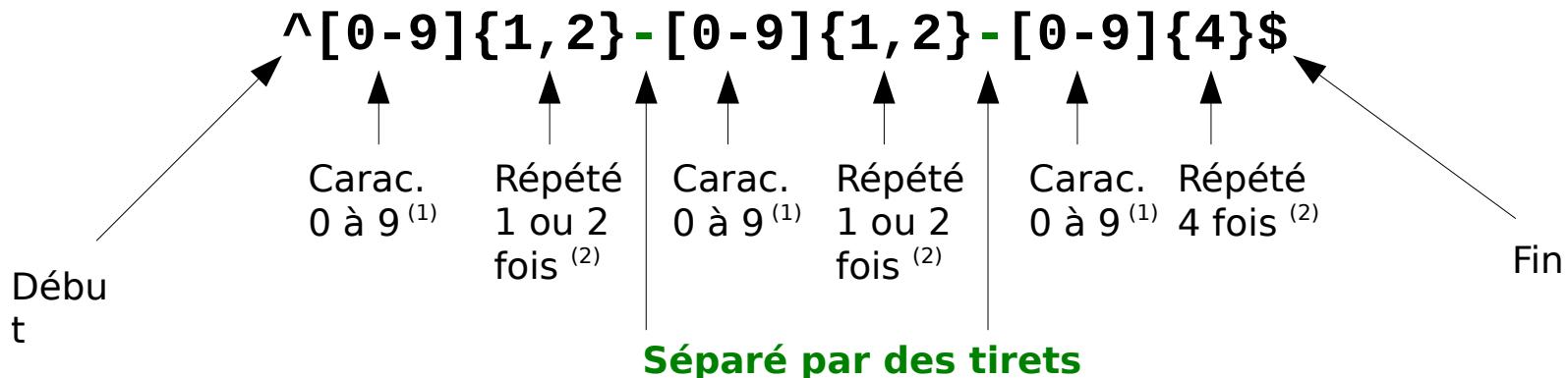


# Compléments et rappels

Sous Linux, beaucoup de fichiers de configuration de services utilisent des "expressions régulières".

## Les expressions régulières: L'essentiel

Exemple: Contrôle de validité d'une date



(1) `.` => N'importe quel caractère

(2) `*` => 0, 1 ou n fois

`+` => 1 ou n fois

`?` => 0 ou 1 fois



# Compléments et rappels

## Les expressions régulières: L'essentiel (suite)

Test sous Linux: Contrôle de validité d'une date

```
# echo "12-11-1978" | grep -E  "[0-9]{1,2}-[0-9]{1,2}-[0-9]{4}$"
12-11-1978
#
```

Autre exemple: Masque pour les fichiers d'extension .gif ou .jpg ou .bmp

$\wedge .^*\backslash.(gif|jpg|bmp)\$$

↑

**Caractère neutralisateur**

Test sous Linux

```
# echo "isat.gif" | grep -E  ".^*\backslash.(gif|jpg|bmp)"
isat.gif
#
```



# Compléments et rappels

## Atelier

- **Configuration au boot**

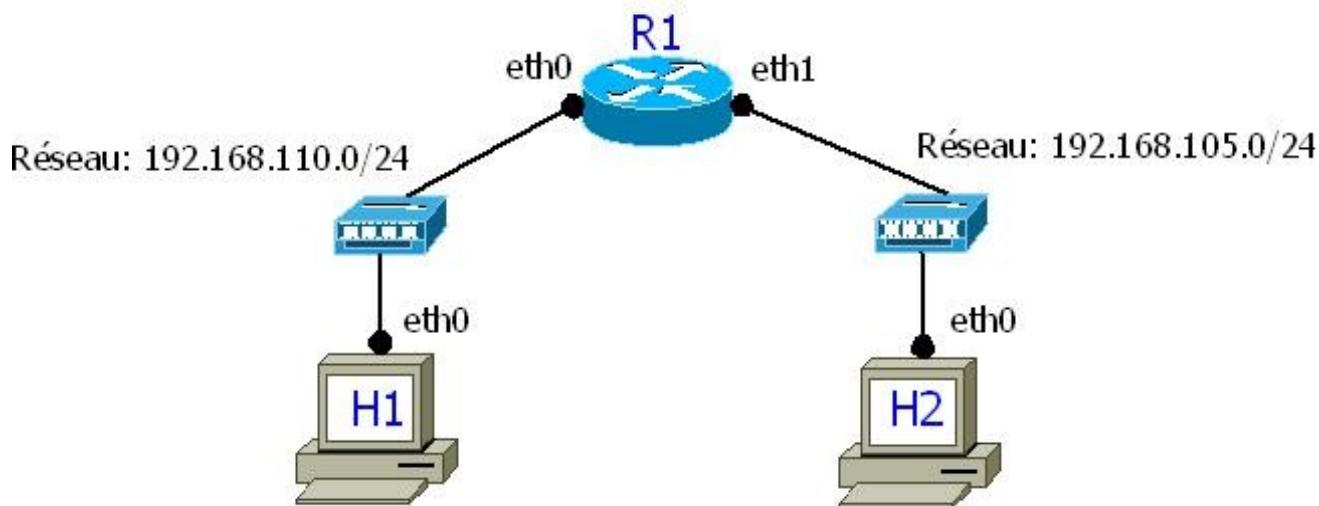
- Quel est le nom du script de la configuration réseau déclenché lors du bootage ?
- Comment démarrer / redémarrer / arrêter l'accès réseau ?
- Après analyse du contenu de ce fichier, trouvez le nom du fichier qui nous permettra d'ajouter des routes statiques lors du bootage ?
- Comment configurer ce fichier ?



# Exercice 1

- **Constituez un petit réseau de 3 machines**

Voici son schéma physique:



R1 joue le rôle de passerelle entre les 2 autres et sera configurée dynamiquement.

Pour H1, la table de routage se construira automatiquement au démarrage.

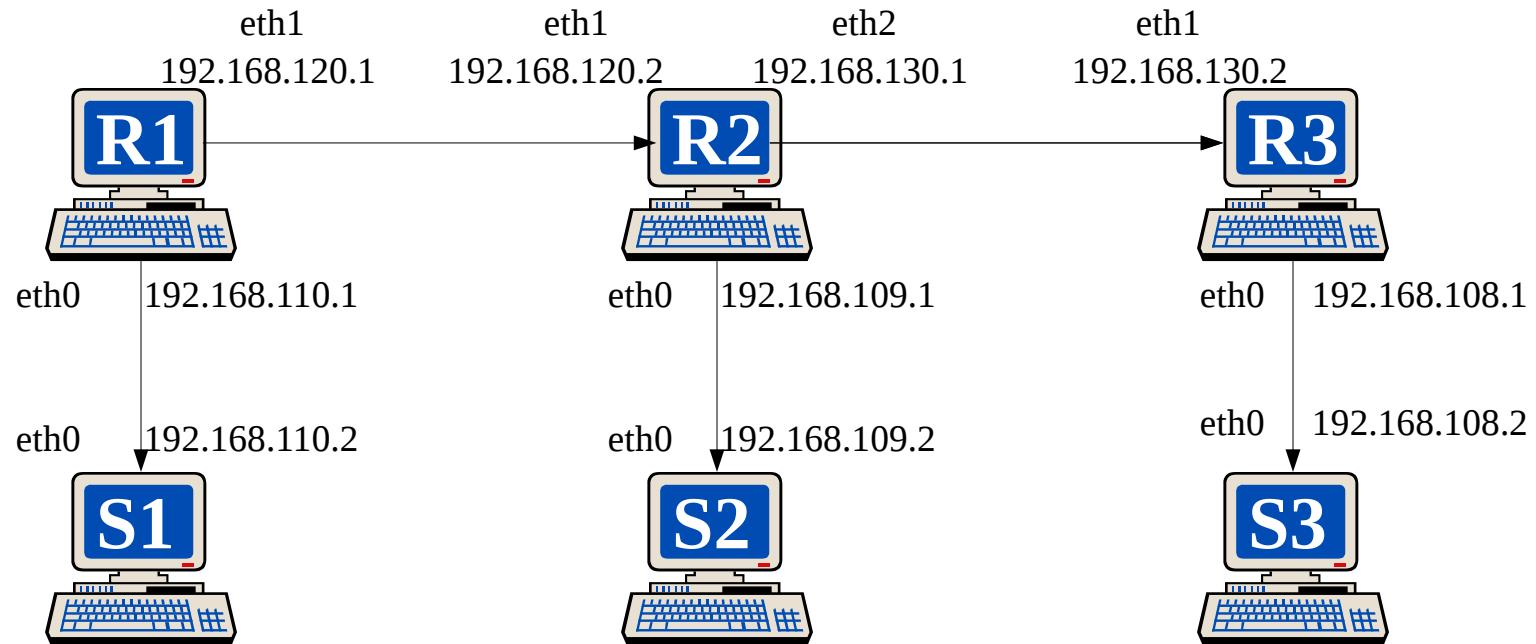
Pour H2, la table de routage sera configurée dynamiquement.

R1, H1 et H2 sont des machines Linux.



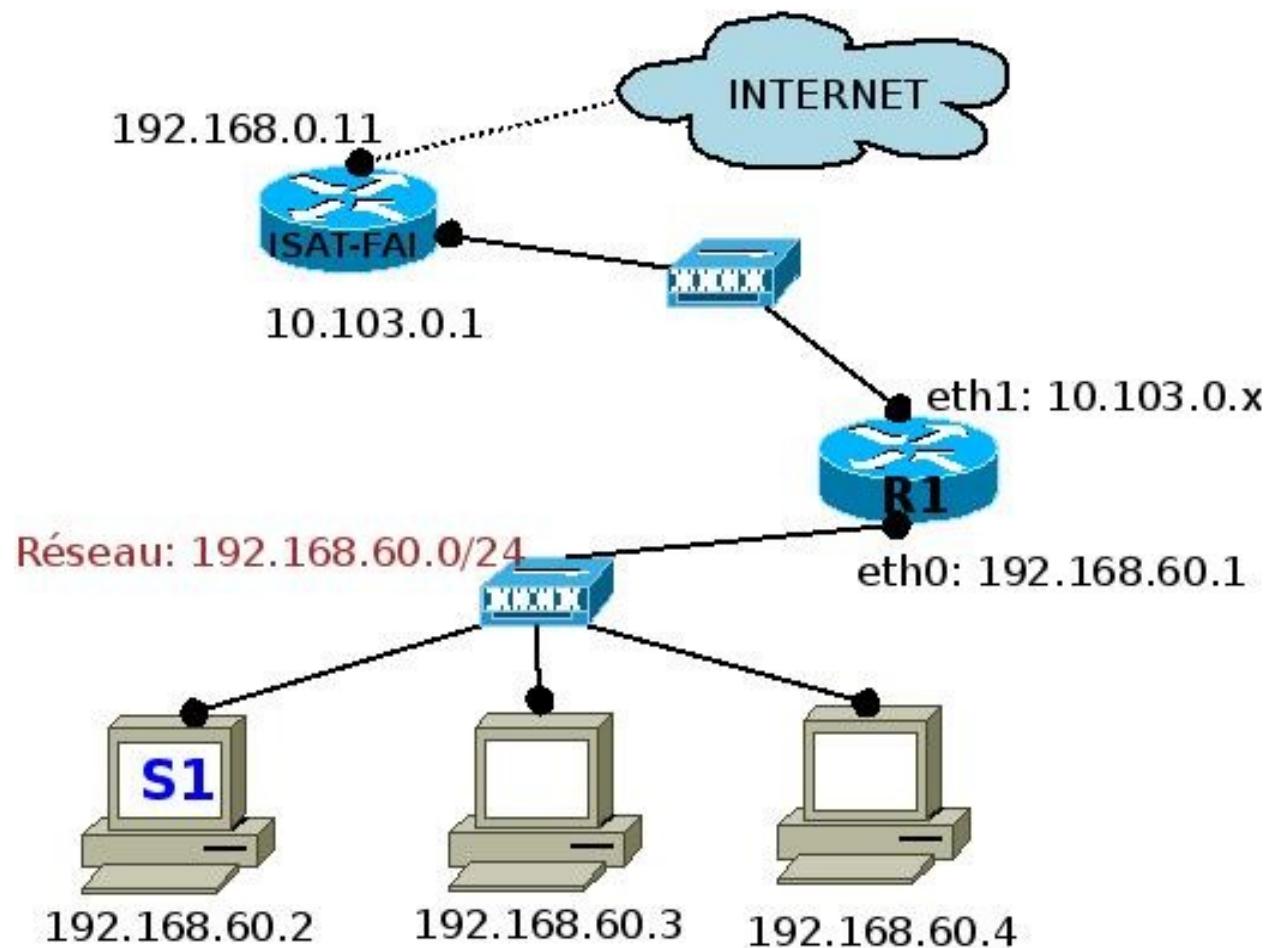
# Exercice 2

- Configurez et testez la maquette suivante (6 machines)**



# Exercice 3

Soit la maquette suivante:



## Exercice 3 (suite)

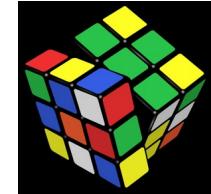
En sachant que ISAT-FAI fait office de routeur d'accès Internet pour le laboratoire:

- a) Quelles sont la (ou les) fonctionnalité(s) à activer sur R1 ?
  
- b) Comment les activer ?
  
- c) Configurez tout le réseau (uniquement S1 et R1) en aval de ISAT-FAI pour que chaque machine puisse accéder à l'Internet.



## Aspects Réseaux sous Linux

**RESOLUTION  
LOCALE DE NOMS**



**Jean-Louis Gouwy**



# Plan

- La résolution de noms
- La résolution locale
- Exercice
- Références



# La résolution de noms

- Pour établir une correspondance compréhensible par les systèmes entre des noms de machines et leur adresse.

Ex: DNS, WINS, NIS ...

- La correspondance peut être connue localement (statiquement) ou être disponible sur le réseau.
- Résolution locale (statique) => /etc/hosts
  - ☺ Un fichier /etc/hosts à gérer pour chacune des machines du réseau
- Solution plus générale (locale + globale - internet) => système DNS (voir plus loin)
  - ☺ Un serveur DNS à configurer sur une seule machine



# La résolution locale

## **1<sup>ère</sup> étape: Choisir le service de résolution de noms**

**/etc/nsswitch.conf** <sup>(1)</sup> (extrait / prioritaire si les 2 existent)

```
...  
hosts: files  nis  dns  
...  
.....
```

<sup>(1)</sup> Prise en compte immédiate des modifications apportées ...



# La résolution locale

## **2<sup>ème</sup> étape: Configurer le(s) service(s) choisi(s)**

files: **/etc/hosts<sup>(1)</sup>**

```
127.0.0.1      theti  localhost.localdomain  localdomain
198.197.56.141 theti  theti.isat.be    papyrus
198.197.56.9   mapasserelle
198.197.56.67  fileserver
```

dns: **/etc/resolv.conf<sup>(1)</sup>**

```
nameserver 193.190.156.67
nameserver 193.190.159.19
```

<sup>(1)</sup> Prise en compte immédiate des modifications apportées ...



# La résolution locale

## - commandes utiles

**hostname** pour montrer ou changer le nom de la machine

**uname -n** pour montrer le nom de la machine

## Les manuels

*man host.conf*

*man resolv.conf*

*man nsswitch.conf*

## - fichiers de configuration

**/etc/nsswitch.conf**

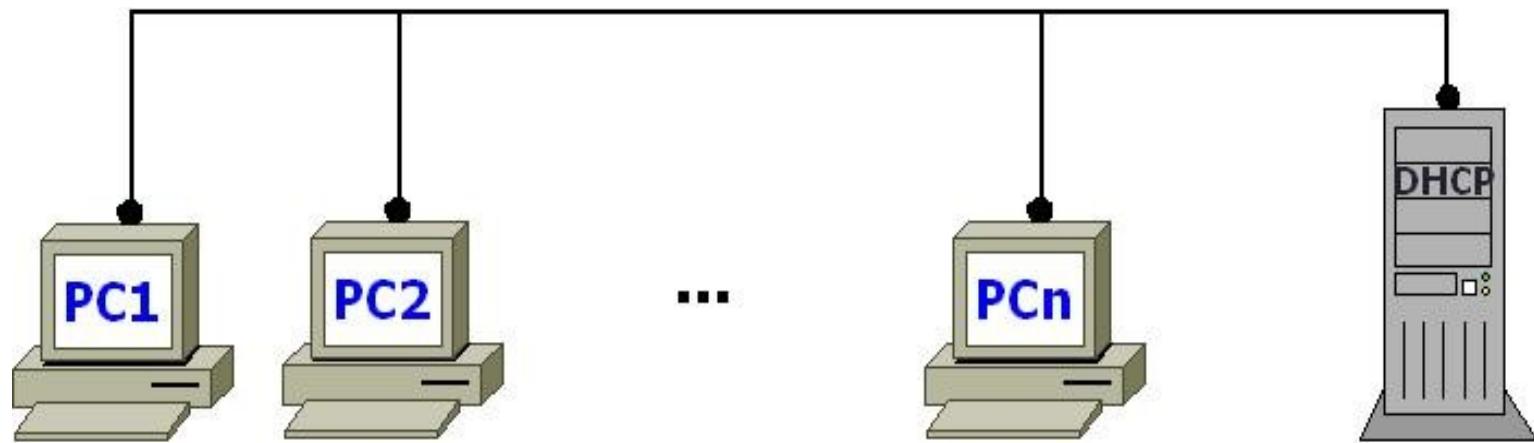
**/etc/hosts**

**/etc/resolv.conf**



# Exercice

- Reconfigurez vos machines pour qu'elles deviennent clientes du serveur DHCP du laboratoire et changez le nom de votre machine en respectant le modèle suivant:



# Exercice (suite)

- Configurez votre fichier 'hosts' local pour pouvoir toucher toutes les machines du réseau.
- Vérifiez la configuration du resolver.
- Pingez toutes les machines du réseau par leur nom et non plus par leur adresse IP.
- Enlevez la tentative de résolution de noms par 'files' dans /etc/nsswitch.conf. Repingez les machines par leur nom.

Que se passe-t-il ?



# Références

## BIBLIOGRAPHIE

ADMINISTRATION UNIX: ASPECTS RESEAUX

Par Xavier Bogaert

Technofutur3

LINUX Red Hat Fedora

Par Bill Ball & Hoyt Duff

CampusPress (Paris)





## Aspects Réseaux sous Linux

# SERVEUR DHCP



Jean-Louis Gouwy



# Plan

- Qu'est-ce que DHCP ?
- Le protocole DHCP
- Fonctionnement
- Atelier
- Le serveur DHCP
- Le client DHCP
- Exercices
- L'agent relais DHCP
- Exercice
- Références



# Qu'est-ce que DHCP ?

- **DHCP**

- Dynamic Host Configuration Protocol

- **OBJECTIFS**

- Protocole permettant d'attribuer dynamiquement une configuration IP aux machines clientes (au minimum: @IP / Netmask)
- Les @IP sont prises dans une plage spécifiée ou fixées en dur.
- Elles peuvent être attribuées durant un certain temps au delà duquel le client devra refaire une requête.

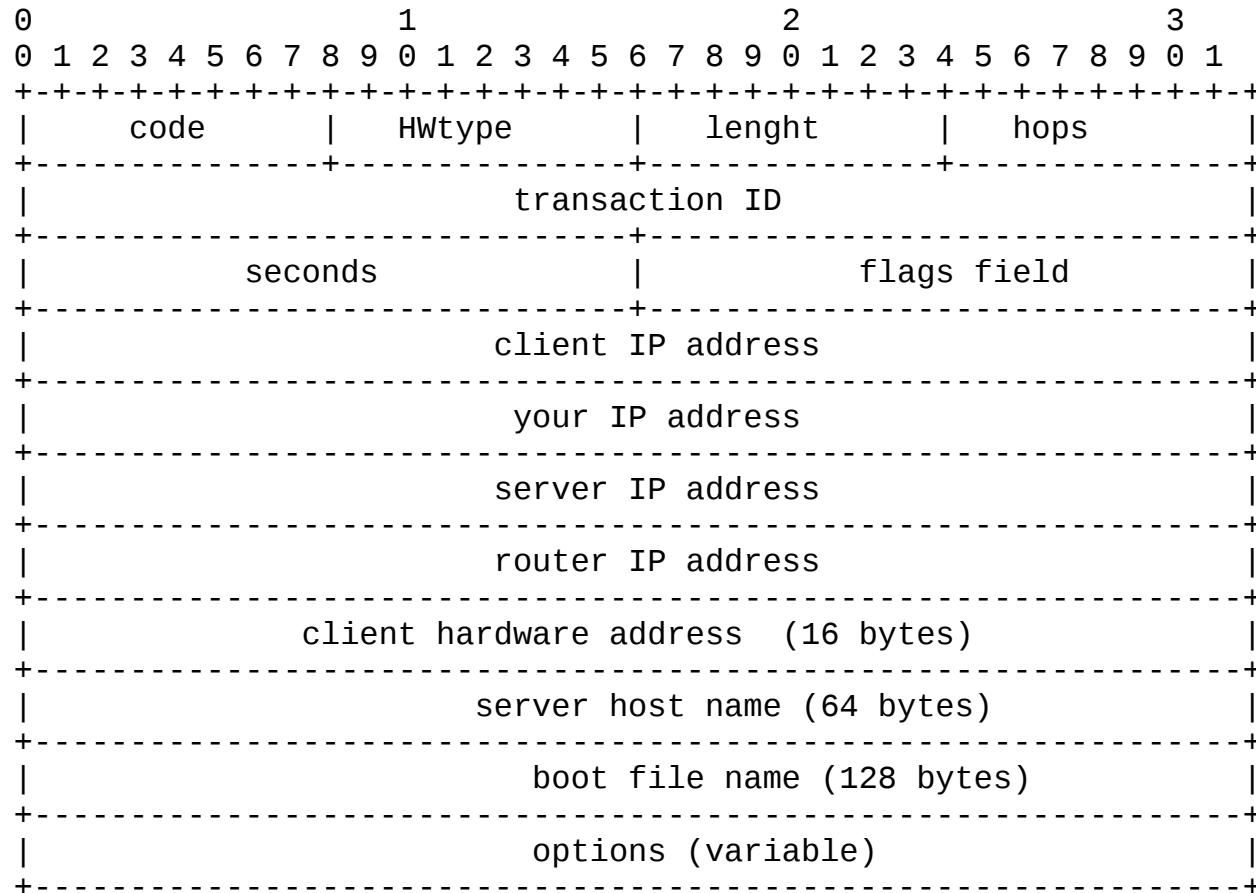
- **INTERETS**

- Centralisation de la configuration des interfaces des postes.
- Pour les réseaux à topologie très variables (portables ...)



# Le protocole DHCP

## DHCP Message Format (RFC 2131)



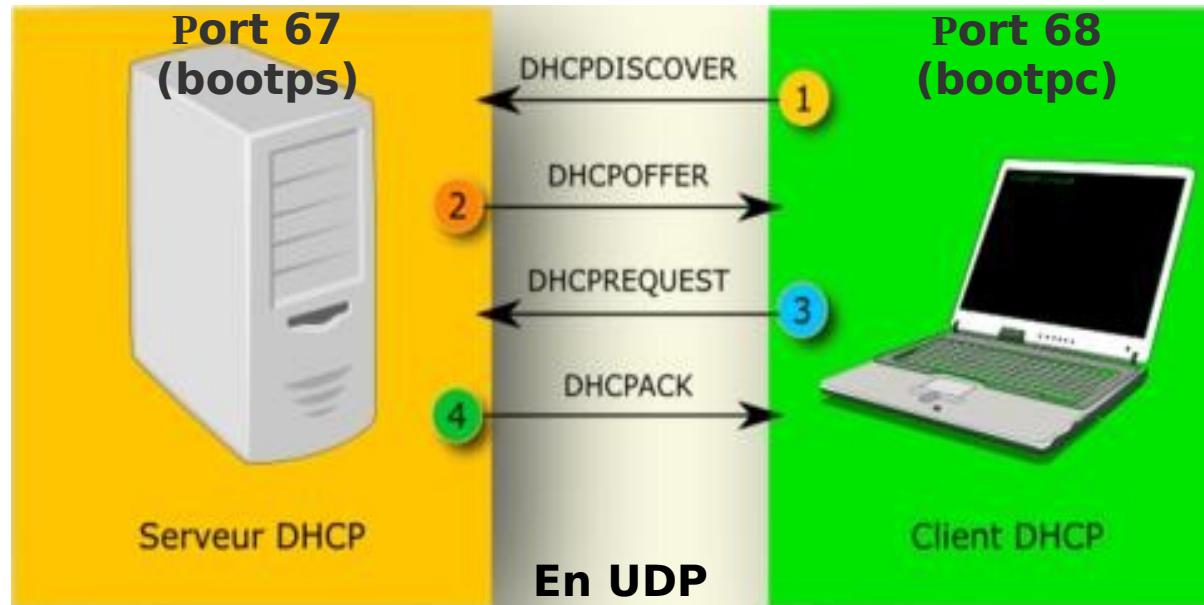
Plus d'info: <http://ylescop.free.fr/mrim/protocoles/rfc-fr/rfc2131.htm>



# Fonctionnement

Lorsqu'un client DHCP n'a encore aucune connaissance du réseau et qu'un serveur DHCP est disponible, le mécanisme classique d'une attribution d'une configuration IP s'articule autour de l'échange de 4 trames:

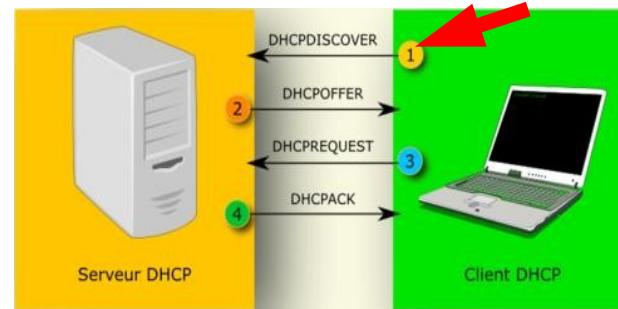
**DHCP DISCOVER -> DHCP OFFER -> DHCPREQUEST-> DHCPACK**



# Fonctionnement (suite)

## 1. DHCP DISCOVER

Un message de *découverte d'un serveur* est envoyé en broadcast Ethernet sur le LAN et est destiné à trouver un serveur DHCP disponible.



### Contenu:

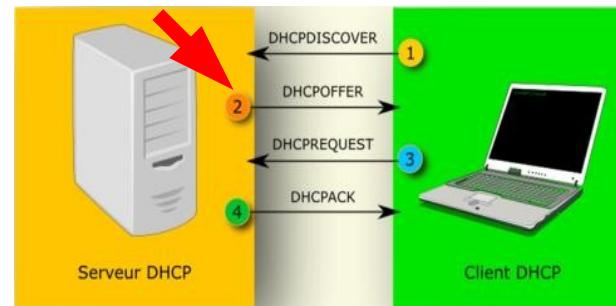
- **@IP client:** 0.0.0.0 (le client n'est pas encore configuré)
- **@Mac client:** le serveur pourra retoucher par la suite le client via cette seule adresse.
- **N° transaction:** pour identifier la transaction (car plusieurs transactions sont possibles en même temps).
- ...



# Fonctionnement (suite)

## 2. DHCP OFFER

Le(s) serveur(s) répondent en émettant un message *d'offre de bail* en broadcast Ethernet ou non selon le type de serveur.



### Contenu:

- **Proposition de configuration:**

- . une @IP
- . un bail
- . etc

- **@Mac client:** pour pouvoir toucher le client concerné.
- **@IP serveur:** si la proposition est acceptée, le client mémorisera l'@IP du serveur.

- **N° transaction**

- ...

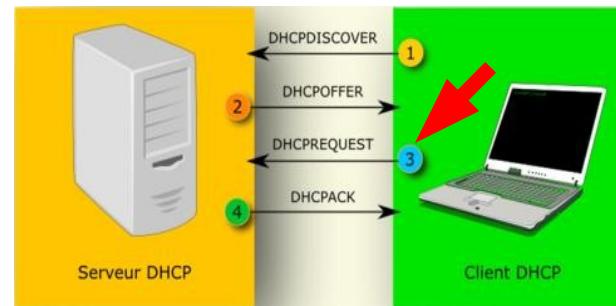


# Fonctionnement (suite)

## 3. DHCP REQUEST

Le client envoie alors *son choix* à tous les serveurs et donc toujours en broadcast.

Ceci pour indiquer l'offre qu'il accepte (généralement la 1ère reçue).



### Contenu:

- **@IP serveur:** c'est l'@Ip du serveur retenu.
- **N° transaction**
- ...



# Fonctionnement (suite)

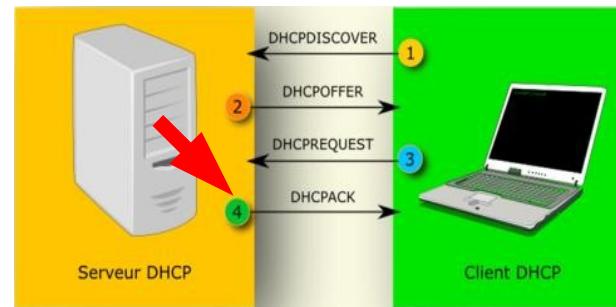
## 4. DHCP ACK

Le serveur concerné répond en unicast définitivement par un accusé de réception qui constitue une confirmation du bail.

L'adresse du client est alors marquée comme utilisée et ne sera plus proposée à un autre client pour toute la durée du bail.

+

Mise à jour des BD d'attribution des bails (côté client et côté serveur)



### Contenu:

- **Paramètres de configuration IP:** qui seront utilisés par le client pour configurer l'interface.
- **Bail et indicateurs de renouvellement**
- **Autres paramètres:** @IPPasserelle, @IPDns, @IPWins...
- **N° transaction**
- ...



# Fonctionnement (suite)

## AUTRES MESSAGES

- |             |  |
|-------------|--|
| DHCPOFFER   | Srv -> Client pour proposer une configuration IP.  |
| DHCPOACK    | Srv -> Client pour signaler que l'adresse IP demandée a été réassignée (bail expiré) ou qu'elle n'est plus actuellement valide (le client a été physiquement déplacé sur un autre réseau). |
| DHCPRELEASE | Client -> Srv pour signaler qu'il libère sa configuration IP et annule son bail.   |
| DHCPIINFORM | Client -> Srv pour recevoir le reste de sa configuration alors qu'il dispose déjà d'une @IP (configurée manuellement par exemple).   |
| DHCPDECLINE | Client -> Srv pour refuser l'offre proposée (l'@IP offerte est déjà utilisée par une autre machine).   |



# Atelier

Analysons maintenant, par un renifleur, une capture de trames correspondant à un dialogue initial DHCP.

## Modèle de travail:

- Lançons un renifleur sur la machine serveur DHCP:

```
# tshark -i ethx port 67 > sniffdhcp.txt
```

- Libérons l'adresse sur la machine cliente:

Prendre la commande qui a permis de lancer le client et remplacer les options "-l" et "-q" par l'option "-r" (release).

```
# /sbin/dhclient -r -lf /var/lib/dhclient/dhclient-eth0.leases  
-pf /var/run/dhclient-eth0.pid eth0
```

- Renouvelons l'adresse sur la machine cliente:

Prendre la commande qui a permis de libérer l'adresse et supprimer l'option "-r".

```
# /sbin/dhclient -lf /var/lib/dhclient/dhclient-eth0.leases  
-pf var/run/dhclient-eth0.pid eth0
```

ou

```
# service network restart
```



# Atelier

- Sur le serveur: # mcedit sniffdhcp.txt
- Analysons ensemble ...



# Serveur DHCP

L'Internet Software Consortium (<http://www.isc.org>) développe un serveur DHCP pour le monde du logiciel libre. C'est le serveur DHCP le plus répandu, et celui qui "suit" au mieux les Rfcs.

- Daemon du serveur DHCP: *dhcpd* (*man dhcpd*)
  - Fichier de configuration: */etc/dhcpd.conf ou /etc/dhcp/dhcpd.conf* (*man dhcpd.conf*)
  - Base des concessions d'@IP: */var/lib/dhcpd/dhcpd.leases* (*man dhcpd.leases*)



# Serveur DHCP (suite)

## LE FICHIER DE CONFIGURATION: **dhcpd.conf**

Il est composé de plusieurs sections, chacune limitée par des accolades.

### Exemple simple

```
# Les options globales (applicables à toutes les sections et redéfinisables)
default-lease-time 259200;   Bail proposé (ici 3 jours)
max-lease-time 518400 ;      Bail maximum proposé si le client est gourmand

subnet 192.168.1.0 netmask 255.255.255.0    Réseau pour lequel dhcpd intervient
{
    range 192.168.1.10 192.168.1.245;          La réserve d'adresses dynamiques
    option subnet-mask 255.255.255.0;           Le masque
    option broadcast-address 192.168.1.255;     Adresse de broadcast
    option routers 192.168.1.1;                  La passerelle par défaut
    option domain-name-servers 192.168.1.6;      Le serveur DNS

host monserveur          Pour le host dont l'@Mac est renseignée, lui attribuer
{                         une adresse fixe.
    hardware ethernet 00:80:C8:85:B5:D2;
    fixed-address 192.168.1.1;    L'adresse fixe ne doit pas appartenir au range !!
}
}
```



# Serveur DHCP (suite)

## BASE DES CONCESSIONS: **dhcpd.leases**

- Doit exister vide (le créer si nécessaire via *touch*) après l'installation.
- C'est la base de données d'attribution des clients DHCP (destinataire, date de début et de fin et l'@Mac de la carte).

### Exemple

```
...
lease 192.168.1.243 {
    starts 1 2010/08/30 15:51:18;
    ends 4 2010/09/02 15:51:18;
    tstp 4 2010/09/02 15:51:18;
    binding state active;
    next binding state free;
    hardware ethernet 00:22:15:56:b7:32;
    uid "\001\000\""\025\226\24TJ";
}
...
```

*Le client d'@Mac 00:22:15:56:b7:32 a reçu l'adresse 192.168.1.243 et ce pour 3 jours.*

### Remarques:

- Il s'agit de l'heure universelle GMT.
- 0: dimanche, 1: lundi .. 6: samedi.



# Serveur DHCP (suite)

## LES INTERFACES D'ECOUTE

Dans le fichier /etc/sysconfig/dhcpd, mettre à jour la variable DHCPDARGS.  
(ex. DHCPDARGS="eth2 eth3 eth4")

## LANCLEMENT/ARRET/REDEMARRAGE

```
# service dhcpcd start/stop/restart
```

## REMARQUES

- Bien souvent un seul serveur DHCP par LAN.
- Le client dhcp peut changer la configuration du fichier /etc/resolv.conf.



# Client DHCP

- Son rôle est de rechercher sur le réseau un serveur DHCP et de négocier avec lui une configuration IP cohérente.
- Le plus en vogue est *dhclient* (*man dhclient*) mais il en existe bien d'autres (dhcpcd, pump, dhcpxd ...).
- Chaque client possède son propre fichier de configuration: dhclient.conf, pump.conf , ... (hors cadre du cours).



# Client DHCP (suite)

**BAILS OBTENUS: /var/lib/dhclient/dhclient.eth<sub>x</sub>.leases**

## Exemple

```
lease {
    interface "eth0";
    fixed-address 192.168.1.243;
    option subnet-mask 255.255.255.0;
    option routers 192.168.1.1;
    option dhcp-lease-time 259200;
    option domain-name-servers 192.168.1.6;

    option dhcp-server-identifier 192.168.1.2;           Serveur DHCP d'origine

    renew 3 2010/09/01 3:51:47;                         1er essai de renouvellement d'adresse (~ ½ du bail)

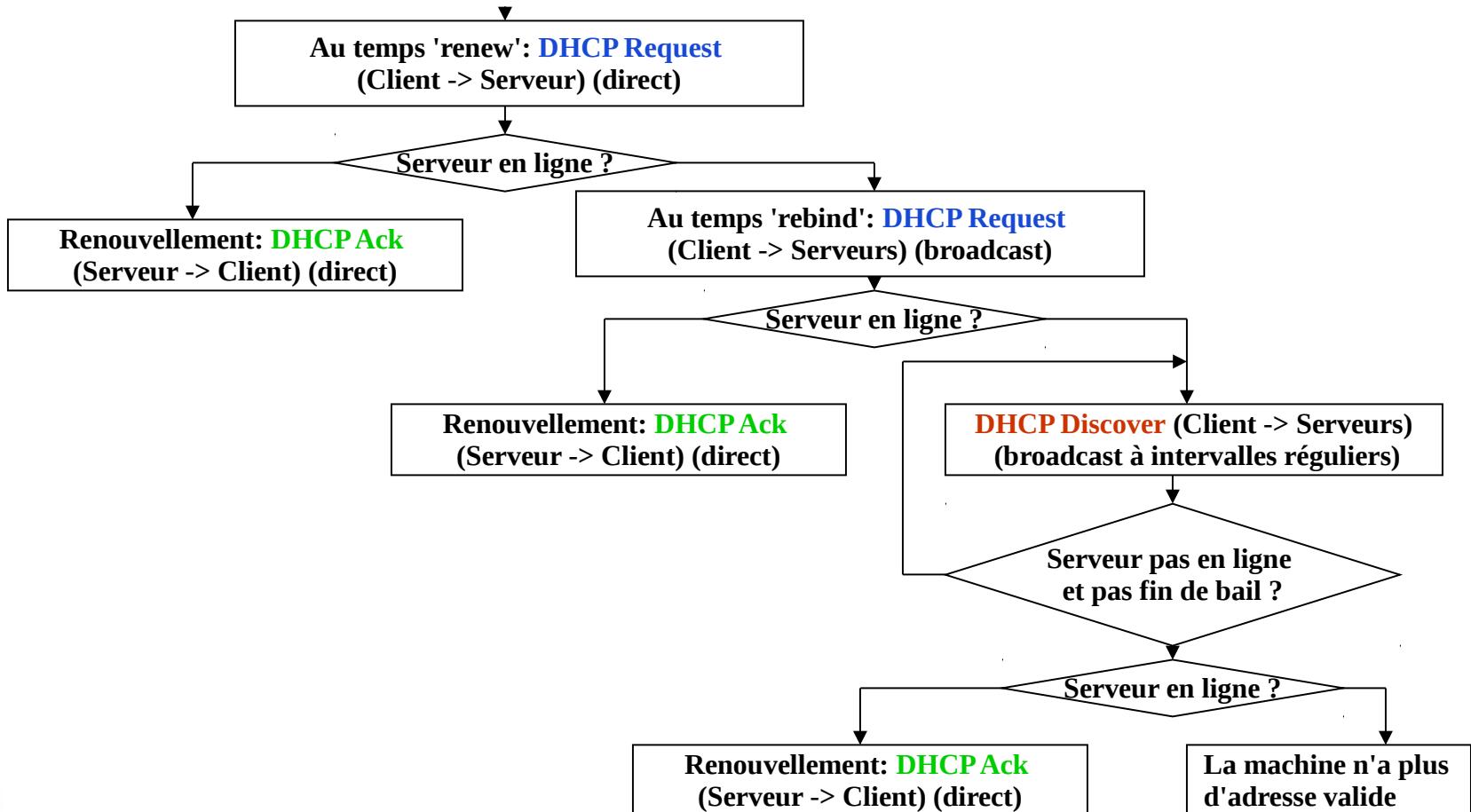
    rebind 3 2010/09/01 21:05:37;                      2eme essai de renouvellement d'adresse (~¾ du bail)

    expire 4 2010/09/02 6:51:18;                      Expiration du bail (~7/8 du bail)
}
```



# Client DHCP (suite)

## MECANISME DE RENOUVELLEMENT DU BAIL (Suite)



# Client DHCP (suite)

## REMARQUES

- Grâce aux informations conservées dans ce fichier `dhclient.leases`, le client *dhclient* adopte un comportement un peu particulier, que l'on ne retrouve pas dans celui de Microsoft, par exemple.

Lorsqu'un hôte a obtenu un premier bail de la part du DHCP, l'adresse du serveur DHCP est conservée et, même après extinction et redémarrage de l'hôte au bout d'un temps bien supérieur à la durée de son bail, le client commencera par envoyer directement un DHCP request au serveur qu'il connaît.

- Sous RedHat, une configuration dynamique d'adresses IP entraîne automatiquement le lancement du client *dhclient*. Ce qui n'est pas le cas pour une configuration statique.



# Client DHCP (suite)

## REFLEXION

Peut-on avoir un client et un serveur DHCP qui tournent sur la même machine ? Oui ou non et pourquoi ?



# Exercices

- **Constituez un petit réseau indépendant de 3 machines**
  - une des machines jouera le rôle de serveur DHCP
  - les 2 autres seront des clientes
- **Installation**
  - installez le serveur DHCP sur la machine serveur
  - installez, si nécessaire, le client DHCP sur les 2 autres machines
- **Configuration de la machine serveur**
  - lui attribuer une adresse IP statique (192.168.0.1/24)
  - configurez le serveur DHCP
    - . plage d'adresses 192.168.0.10 à 192.168.0.20
    - . une adresse fixe (192.168.0.21) pour une des deux machines
    - . un bail de 4 heures pour la première machine et un de 8 heures pour la seconde
    - . attribution automatique d'un hostname à la machine d'adresse fixe (voir 'option host-name' ou 'use-host-decl-names')



# Exercices (suite)

- **Configuration de la 1ère machine cliente**

- attribution dynamique de ses paramètres IP
- vérifiez que le serveur DHCP ne tourne pas
- lancez, si nécessaire, le client DHCP (s'il ne tourne pas déjà)
- redémarrez le réseau
- vérifiez si l'interface est correctement configurée

- **Configuration de la 2ème machine cliente**

- Idem 1ère machine



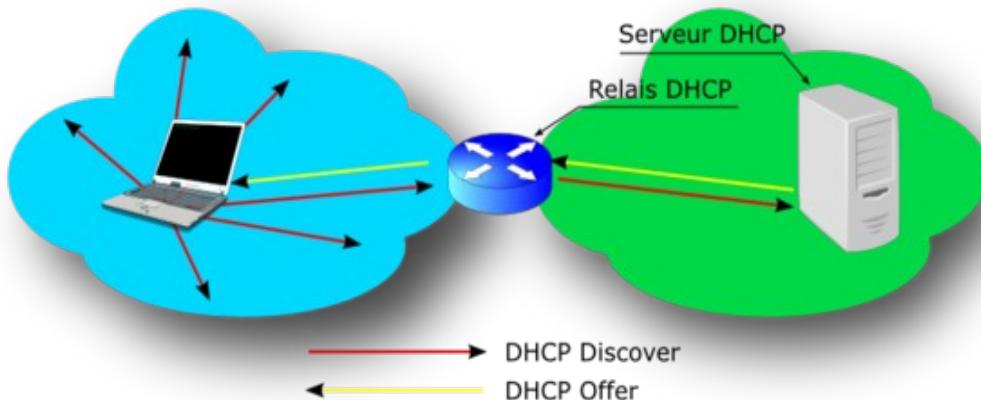
# Exercices (suite)

- Comment connaître l'@Mac d'une interface distante à l'aide d'un serveur DHCP ?
- Examinez et vérifiez le fichier dhcpd.leases et dhclient.eth<sub>x</sub>.leases après une connexion d'un client.
- Examinez et vérifiez le fichier dhcpd.leases après une déconnexion d'un client.
- Examinez et vérifiez le fichier dhcpd.leases après la libération de l'adresse par le client.
- Comment tester qu'un client redemande une adresse en fin de bail ? Faites l'expérience au laboratoire.



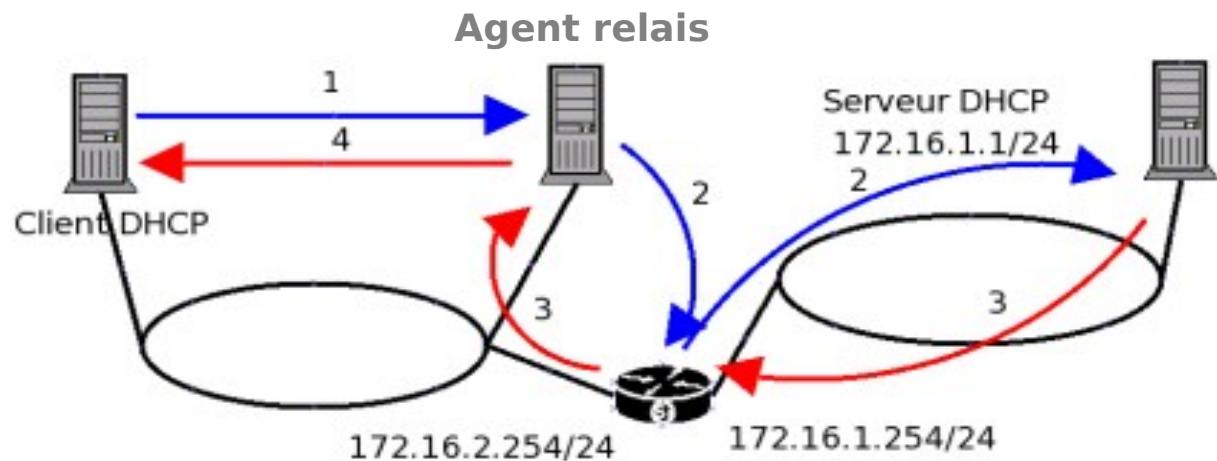
# L'agent relais DHCP

- Comme les clients contactent les serveurs DHCP à l'aide d'une diffusion, dans un inter-réseau, vous devriez théoriquement installer un serveur DHCP par sous-réseau.
- C'est ici qu'intervient l'agent relais DHCP dont le rôle est d'intercepter les requêtes en broadcast et les transmettre à un serveur DHCP connu de cet agent.
- Si votre routeur prend en charge la RFC 1542, il peut faire office d'agent de relais DHCP, et ainsi relayer les diffusions de demande d'adresse IP des clients DHCP dans chaque sous-réseau.



# L'agent relais DHCP (suite)

- Dans le cas contraire, une machine peut être configurée comme agent de relais DHCP.

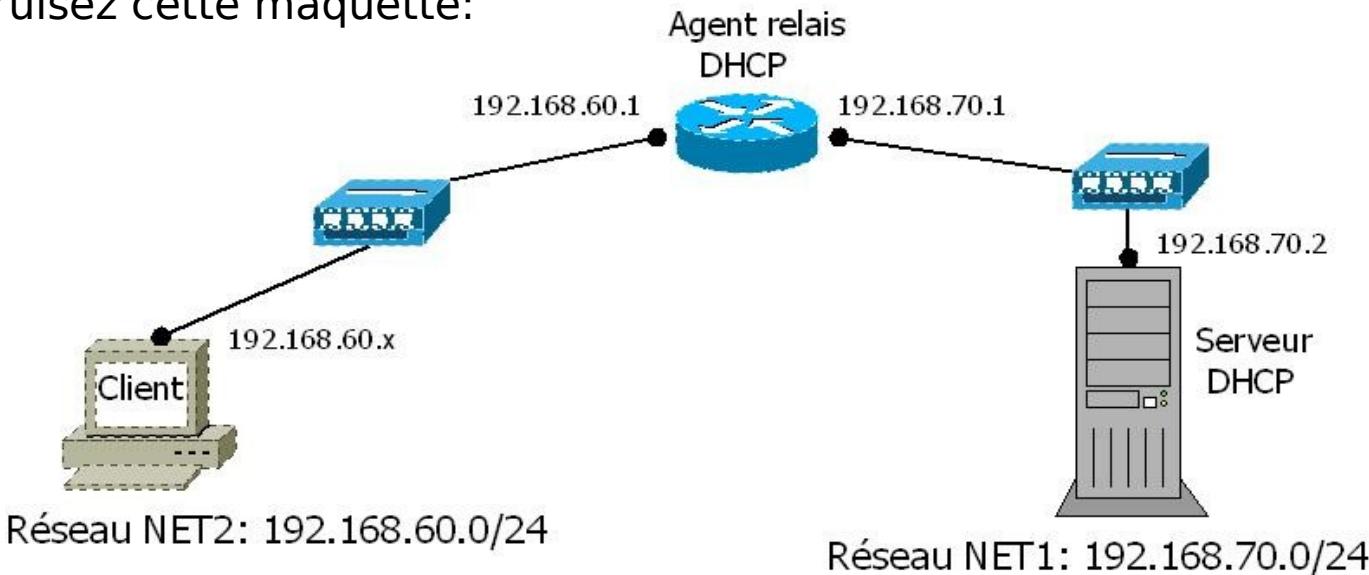


- <sup>(1)</sup> Après avoir envoyé une trame de broadcast, le client DHCP dialogue avec l'agent de relais en unicast.
- <sup>(2)</sup> L'agent demande une configuration IP au serveur DHCP dont il connaît l'adresse.
- <sup>(3)</sup> Le serveur retourne une configuration à l'agent.
- <sup>(4)</sup> Celle-ci est donnée au client DHCP par l'agent.



# Exercice

Construisez cette maquette:



- L'installation du serveur dhcp inclut celle de l'agent relais (dhcrelay).
  - Le serveur dhcp ne doit pas tourner sur l'agent relais.
  - Fichier de configuration de l'agent relais: /etc/sysconfig/dhcrelay
  - Démarrage de l'agent: # service dhcrelay start
  - Activation manuelle de l'agent: commande dhcrelay (*man dhcrelay*)
- En vue de réaliser l'exercice sur 3 machines, nous installons ici l'agent directement sur la passerelle.*



# Références

## WEBOGRAPHIE

<http://www.frameip.com/dhcp>  
<http://christian.caleca.free.fr/dhcp.html>  
<http://www.linux-france.org/prj/edu/archinet/systeme/ch29.html>

## BIBLIOGRAPHIE

TCP/IP

Par Joe Casad & Bob Willsey (Edition CampusPress)

CentOS Bible

By Timothy Boronczyk and Christopher Negus - Edition 2009 - Wiley Publishing, Inc.

Le Campus - Linux: Installation, configuration et applications

Par Michael Kofler - 8ème Edition 2009 - Pearson Education France

The DHCP HandBooks (2ème édition) - Par Ralph Droms & Tedemon

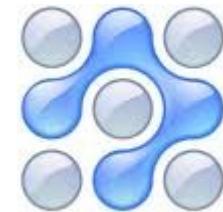
ADMINISTRATION UNIX: ASPECTS RESEAUX Par Xavier Bogaert

Technofutur3



## **Laboratoire de sécurité internet**

# **LE SUPER-SERVICE XINETD**



**Jean-Louis Gouwy**



# Plan

- Présentation
- Configuration
- Exercices
- Références



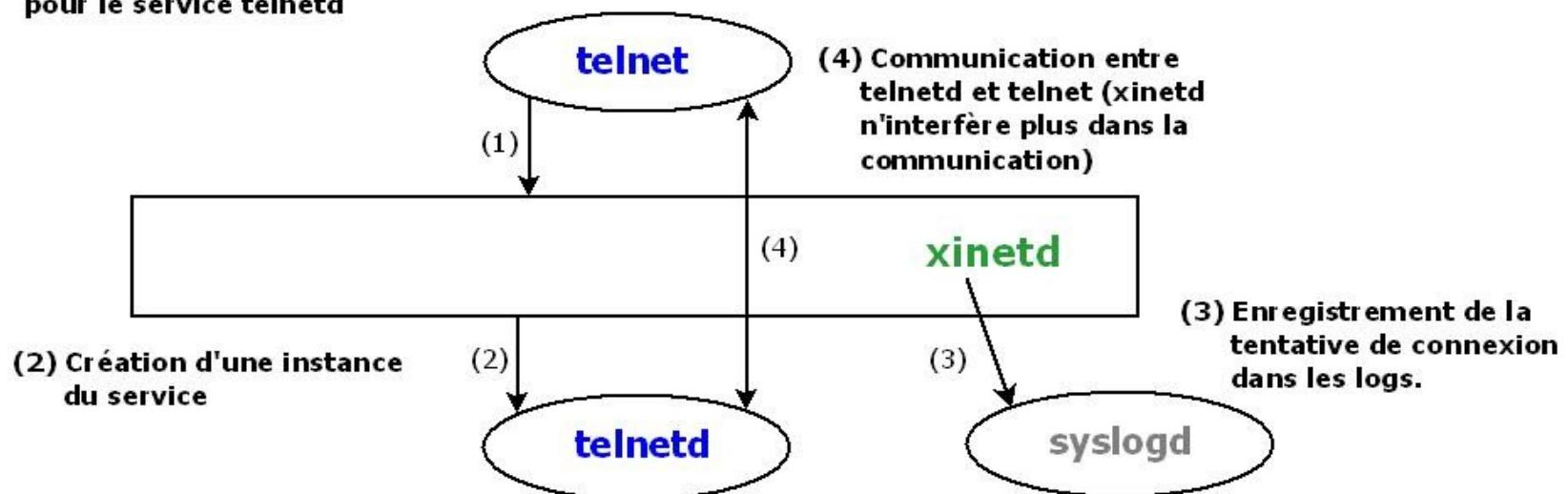
# Présentation

- En général, un service = un daemon qui écoute sur un port particulier.
- ⌚ Les services consomment des ressources même s'ils ne sont que rarement utilisés.

Idée: Un super-service (xinetd - extented Internet services daemon) qui écoute sur une série de ports et qui lance le daemon adéquat lorsqu'une connexion arrive. De plus, il fournit une excellente sécurité contre les intrusions, et limite certains risques d'attaque par *Deny of Services* (*DoS*).

## (1) Requête telnet

+ Vérification des règles xinetd définies pour le service telnetd



Xinetd (JL Gouwy)



# Présentation (suite)

- Beaucoup de services réseaux (pas tous) peuvent être configurés pour fonctionner avec xinetd (ftp, ssh, samba, apache v1 ...)
- Plus riche et plus souple, xinetd remplace son ancêtre 'inetd'.
- Il permet en plus d'autoriser ou non les connexions sur la machine en fonction d'une adresse IP, du nom d'hôte du client ou de son domaine, de l'heure, du taux de charge, du nombre de connexions simultanées, du nombre de connexions entrantes par seconde ...
- Il n'est toutefois pas aussi puissant qu'un vrai firewall.

*man xinetd et xinetd.conf*



# Configuration

## Le fichier /etc/xinetd.conf

Indique la configuration par défaut qui s'applique à tous les services et l'emplacement des fichiers de configuration propre à chaque service de xinetd.

**defaults**

{

**disable = yes** *Tout est désactivé par défaut.*

**instances = 20** *Nombre de requêtes simultanées que xinetd peut gérer par service.*

**no\_access = 0.0.0.0/0** *Par défaut aucun réseau ne peut se connecter.*

**log\_type = SYSLOG authpriv** *Envoyé à syslog comme authpriv.info.*

**log\_on\_failure = HOST RECORD**

*Lorsqu'un serveur ne peut démarrer (soit par manque de ressources, soit à cause des règles d'accès), on enregistre l'adresse du client, et des informations sur le client.*

**log\_on\_success = HOST USERID DURATION PID**

*Enregistre en cas de succès de la connexion, l'adresse du client, l'identité de l'utilisateur distant, la durée de la session, le PID du serveur (s'il s'agit d'un service interne à xinetd, le PID vaut 0)*

**per\_source = 4** *On n'autorise que 4 connexions vers le même service venant de la même machine (prévention contre une attaque de type DoS).*

}

**includedir /etc/xinetd.d** *Répertoire où se trouvent les fichiers de configuration par service.*



# Configuration (suite)

## Les fichiers /etc/xinetd.d/\*

Ils se trouvent dans le répertoire /etc/xinetd.d par défaut. Pour chaque service, il doit y avoir un fichier portant le nom du service. Prenons le contenu du fichier telnet.

```
service telnet
{
    disable = no
    flags = REUSE
    instances = UNLIMITED
    only_from = 172.16.0.0/16
    only_from = .bacisat.be
    only_from = 10.0.0.{10,11,12}
    socket_type = stream
    wait = no
    user = root
    server = /usr/sbin/in.telnetd
}
```

*On suppose ici qu'on veut l'activer.*

*Réutilisation du socket pour une autre connexion telnet.*

*Pas de limitation. Le nombre maximal de serveurs d'un même type pouvant fonctionner en même temps. A éviter >attaque DoS.*

*Connexions acceptées depuis les adresses 172.16.0.0 avec un masque de 255.255.0.0*

*Idem mais depuis des machines du domaine bacisat.be*

*Idem depuis 10.0.0.10, 10.0.0.11, 10.0.0.12*

*Type de service de transport de données (stream pour communication en mode connecté (ex.tcp), dgram pour communication en mode non connecté (ex.udp), raw pour accès direct à IP (ex. icmp).*

*Le service est multi-thread. A chaque nouvelle requête vers le service, un nouveau serveur est démarré par xinetd, dans la limite maximale définie par la clause 'instances' .*

*Nom de l'utilisateur sous lequel le daemon tourne.*

*Chemin d'accès au programme in.telnetd lancé par xinetd (il est possible ici d'ajouter les options de démarrage du programme).*



# Configuration (suite)

On a la possibilité d'ajouter d'autres options, par exemple:

**access\_time = 7:00-12:30 13:30-21:00** *On autorise l'accès durant ces plages horaires.*  
**nice = 10** *Modifie la priorité du processus (idem commande 'nice').*  
**bind = a.b.c.d** *Permet de lier un service à une adresse IP, il a de l'intérêt si vous disposez d'au moins deux adresses IP sur la machine.*  
**redirect = a.b.c.d** *Permet de rediriger la requête vers une autre machine d'adresse a.b.c.d*

Pour finir, certains attributs sont obligatoires pour activer un service:

- socket-type
- user (dans le cas d'un service non interne).
- server (dans le cas d'un service non interne).
- wait
- protocol (le protocole stipulé doit exister dans /etc/protocols - ex. protocol = tcp - S'il n'est pas indiqué, le protocole par défaut associé à ce service est employé).
- port (même numéro de service que celui présent dans /etc/services - ex. port =23).



# Exercice 1

- Si ce n'est pas déjà le cas, installez xinetd
  - vérifiez si le package est installé
  - vérifiez si /usr/sbin/xinetd existe
  - lancez xinetd
  - vérifiez si xinetd est dans la liste des processus
- Service interne: Configurez le service echo (port tcp 7)
  - vérifiez que "telnet localhost 7" fonctionne
  - ne pas oublier de redémarrer xinetd avant ...
- Service externe: Configurez xinetd
  - pour que les accès telnet soient impossible à partir de n'importe qu'elle machine de la salle de formation sauf de la machine de votre voisin (à réaliser par groupe de 2 PC).
  - pour que tous les services inutiles soient bloqués pour tout le monde
  - utilisez les utilitaires que vous connaissez afin de vérifier cela
  - vérifiez dans les logs les résultats obtenus



## Exercice 2

xinetd permet également de créer facilement un petit serveur sans avoir à manipuler les sockets.

- Créez un service `/sbin/coucou.sh` (script écrit en bash) qui écoutera sur le port 50000 et qui respectera le protocole suivant :

Lors de la connexion au service, celui-ci attend l'introduction d'un mot :

Si ce mot vaut :

"bye" alors il retournera "au revoir"  
"hello" alors il retournera "bonjour"  
tout autre chose alors il retournera "pas compris"

... puis réécoutera un nouveau mot

Pour sortir, il faudra saisir "quit"



# Références

## WEBOGRAPHIE

<http://www.xinetd.org/>

<http://www.linuxfocus.org/Francais/November2000/article175.shtml#lindex6>

## BIBLIOGRAPHIE

ADMINISTRATION UNIX: ASPECTS RESEAUX

Par Xavier Bogaert

Technofutur3

PREPARATION A LA CERTIFICATION LPIC-1 LINUX

Par Sébastien Rohaut

Eni Edition 2008



## Aspects Réseaux sous Linux

# OPENSSH



**Jean-Louis Gouwy**



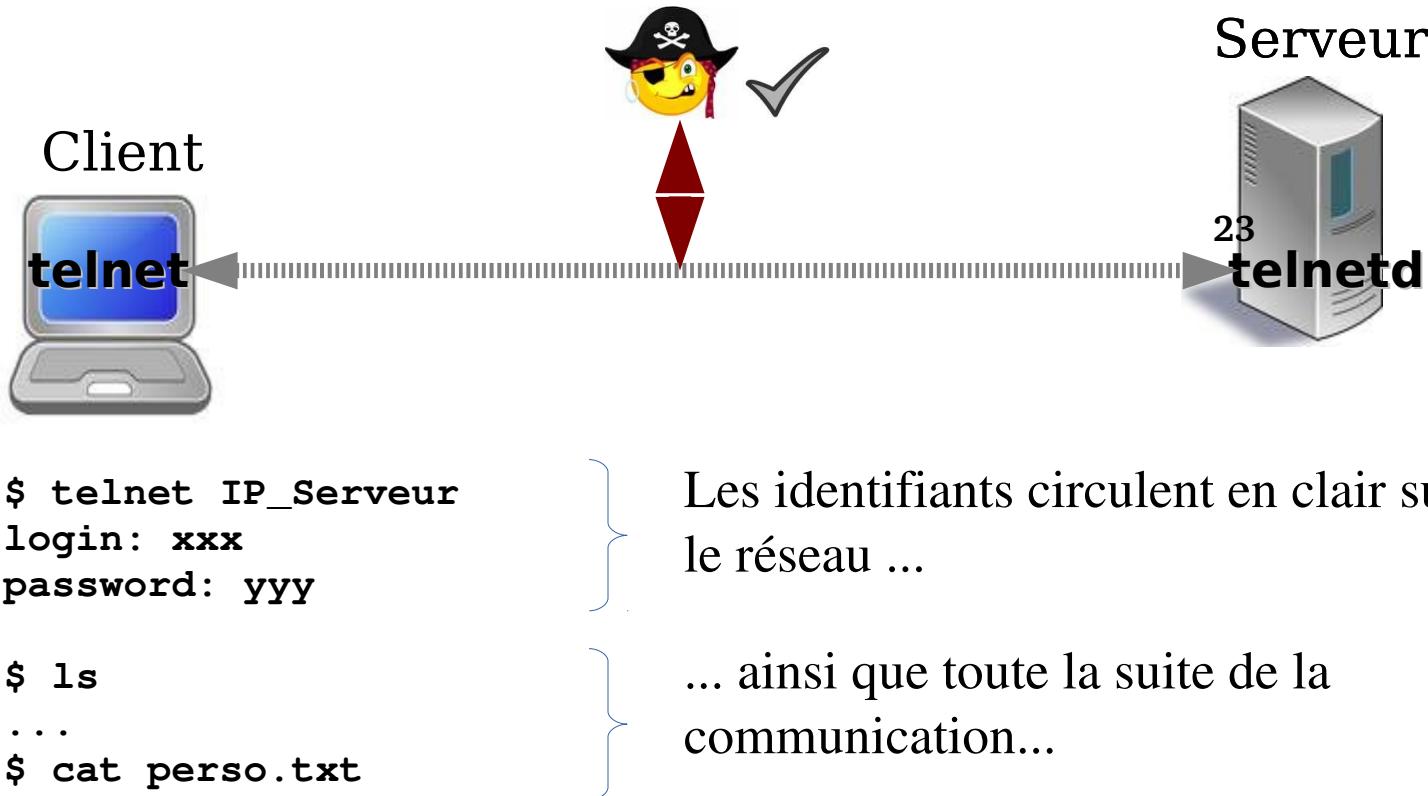
# Plan

- Introduction
- La cryptographie
- Qu'est-ce que OpenSSH ?
  - Utilités et avantages
- Etablissement d'une communication
- Configuration du serveur
- Implémentation par mot de passe
  - Remarques
  - Exercice
- Implémentation par clés
  - Remarques
  - Exercice
- Avantages fonctionnels et techniques
- Gestion d'un parc Linux
  - Implémentation de la solution
  - Exercices
- Quelques outils
- Le port forwarding
  - Exercice
- Le tunneling
- Références



# Introduction

- **Une conversation non-sécurisée: telnet**



# Introduction

- **Atelier: Espionner une conversation telnet**

- Lancer *Wireshark* sur le poste client.
- Définir un filtre de capture pour telnet (*tcp.port==23*) et pour l'interface concernée.
- Lancer la capture.
- Etablir une connexion telnet.
- Analyser le contenu des trames en vue d'y retrouver les identifiants.



# La cryptographie

- **Définition**

Regroupe l'ensemble des techniques permettant de (dé)chiffrer des messages.

## Vocabulaire:

Déchiffrer: Obtenir le message en clair à l'aide de la technique prévue.

Décrypter: Obtenir le message en clair à l'aide de techniques de piratage.



# La cryptographie

- **Les algorithmes de chiffrement**



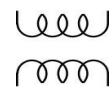
Les données sont toujours  
(dé)chiffrées de la même façon.

## Exemples:

- . Chiffrement de César
- . Chiffrement par décalage de bits
- . etc ...



Message en clair (*Plaintext*)



Algorithme de (dé)chiffrement



Cryptogramme (*Ciphertext*)



# La cryptographie

- **Les algorithmes de chiffrement symétriques**



Les données sont (dé)chiffrées selon un algorithme paramétré par une *clé secrète (clé de chiffrement)*.

La clé secrète est la même au chiffrement et au déchiffrement.



La clé secrète



# La cryptographie

- **Les algorithmes de chiffrement symétriques**

[https://fr.wikipedia.org/wiki/Catégorie:Algorithme\\_de\\_cryptographie\\_symétrique](https://fr.wikipedia.org/wiki/Catégorie:Algorithme_de_cryptographie_symétrique)

## Exemples:

- . Chiffrement de César (la clé = la valeur du décalage).
- . Chiffrement par décalage de bits (la clé = la valeur du décalage)
- . Chiffrement par substitution (la clé = le tableau de substitution).
- . Chiffrement de Vigenère

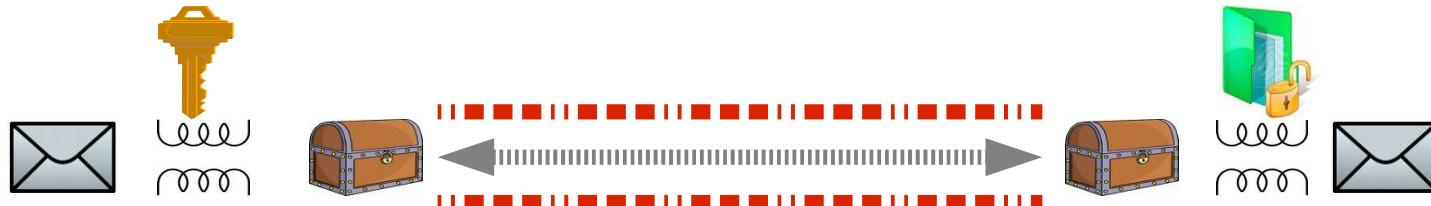
## Beaucoup plus sérieux:

- . **DES** (Data Encryption Standard) → Lent et crackable par force brute
- . **3DES** (TripleDES) → Plus difficilement crackable mais très lent
- . **AES** (Advanced Encryption Standard): remplace le DES et le 3DES trop gourmands en ressources, peu performants et assez vulnérables.
- . **RC5** (Rivest Cipher 5): simple, rapide et facile à mettre en oeuvre.
- . **Blowfish**, etc...



# La cryptographie

- **Les algorithmes de chiffrement asymétriques**



Un trousseau de clés est composé d'une clé publique et d'une clé privée.

Un message chiffré avec une clé publique ne pourra être déchiffré qu'avec la clé privée du trousseau et réciproquement.

Les clés de chiffrement (publiques) et de déchiffrement (privées) sont différentes mais '*complices*' ou '*amies*'.

Il n'est pas possible de retrouver la clé privée par sa clé publique (ou inversément).



Clé publique du destinataire



Clé privée du destinataire



# La cryptographie

- **Les algorithmes de chiffrement asymétriques**

[https://fr.wikipedia.org/wiki/Catégorie:Algorithme\\_de\\_cryptographie\\_asymétrique](https://fr.wikipedia.org/wiki/Catégorie:Algorithme_de_cryptographie_asymétrique)

## Exemples:

- . **RSA:** Rivest Shamir Adleman (1977)

RSA a été breveté par le MIT en 1983 aux États-Unis.

Le brevet a expiré le 21 septembre 2000.

En 2002, il servait encore à protéger les codes nucléaires de l'armée américaine et russe.

Actuellement, c'est le système à clef publique le plus utilisé (carte bancaire, de nombreux sites web commerciaux...).

- . **DSA:** **D**igital **S**ignature **A**lgorithm

Algorithme de signature numérique standardisé par le National Institute of Standards and Technology (NIST) aux États-Unis.

Il peut être utilisé gratuitement.

- . etc ...



# La cryptographie

- **Les algorithmes de chiffrement**

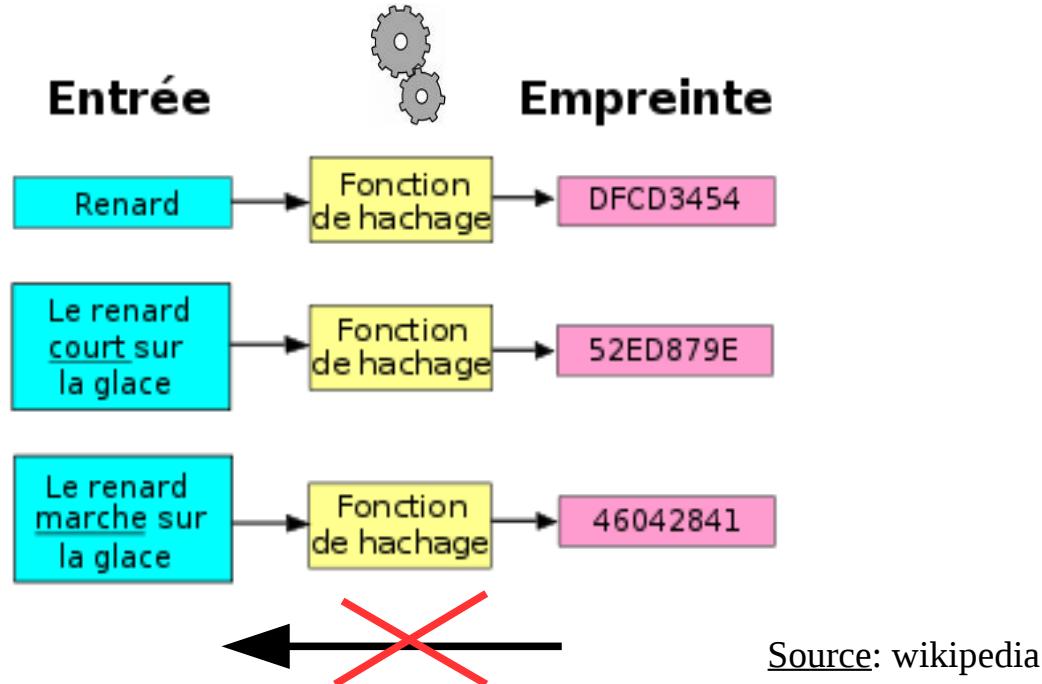
- . Les algorithmes de chiffrement en tout genre permettent de s'assurer que l'information n'est seulement accessible qu'à ceux dont l'accès est autorisé (règle de **confidentialité**).
- . Les chiffrements symétriques sont plus rapides que les chiffrements asymétriques.  
Ils sont souvent utilisés pour chiffrer le transfert réseau.
- . La vitesse de transfert est d'autant plus rapide que la clé de session est courte; mais le degré de confidentialité est d'autant plus élevé que la clé de session est longue.



# La cryptographie

- **Les algorithmes de hachage**

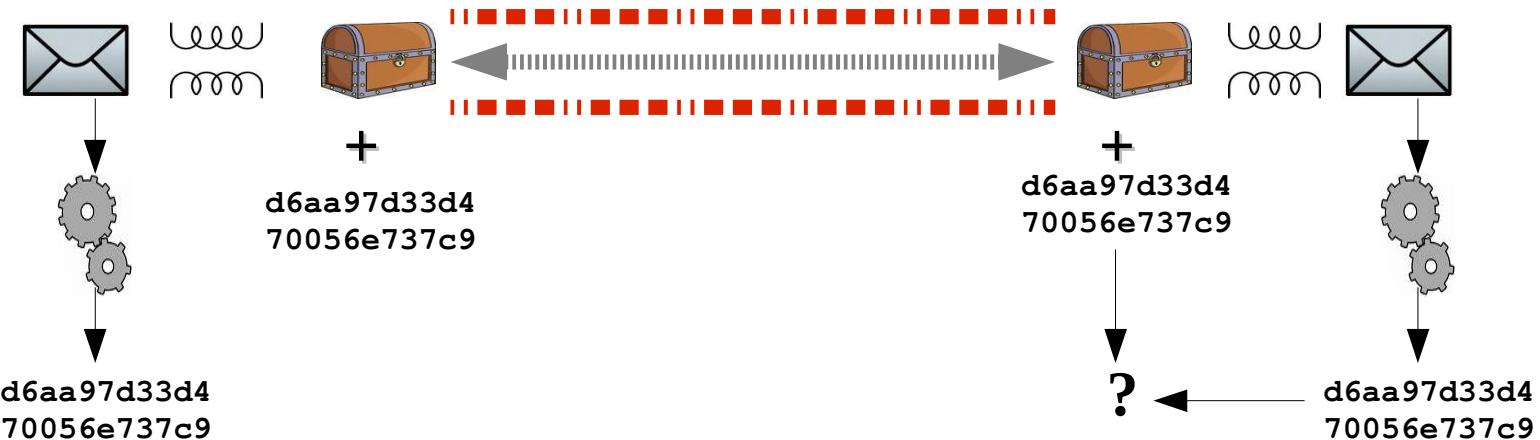
On nomme fonction de hachage une fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une *empreinte* (condensat ou *signature*) servant à identifier rapidement, bien qu'incomplètement, la donnée initiale.



# La cryptographie

- **Les algorithmes de hachage**

- Une empreinte est calculée et envoyée en même temps que le message sur laquelle elle porte.
- Une empreinte est calculée sur le message reçu.
- On peut considérer que le message n'a pas été modifié durant le transfert si l'empreinte recalculée correspond à celle reçue avec le message (règle d'**intégrité**).



# La cryptographie

- **Les algorithmes de hachage**

## Exemples:

- . **MD5** (Message Digest 5): produit un condensat de 128 bits.
  - peut accompagner certains téléchargements  
(vérification: commande `md5sum`)
  - calculer l'empreinte d'un mot de passe (GNU/Linux) avec injection d'un salage pour compliquer le décryptage.
- Rem. Considéré comme obsolète car on a trouvé 2 messages qui génère la même empreinte !*
- . **SHA** (Secure Hash Algorithm): produit un condensat de 160 à 512 bits
  - peut accompagner certains téléchargements  
(vérification: commande `sha1sum`)
- . etc ...



# La cryptographie

- **Ateliers de hachage**

## a. Utilisation de la commande MD5

- Générez une empreinte sur un fichier
- Vérification de cette empreinte sur:
  - a) Ce même fichier non modifié
  - b) Ce même fichier modifié

## b. Vérification de l'intégrité d'un logiciel téléchargé

Vérifiez l'intégrité du package webmin (<http://www.webmin.com>):

1. Téléchargez Webmin (version minimale)
2. Vérification de son empreinte MD5



# La cryptographie

- **Algorithme Diffie-Hellman: La transmission de clefs secrètes**

C'est une méthode par laquelle deux personnes nommées conventionnellement Alice et Bob peuvent se mettre d'accord sur un nombre (qu'ils peuvent utiliser comme clé pour chiffrer la conversation suivante) sans qu'une troisième personne appelée Eve puisse découvrir le nombre en écoutant.

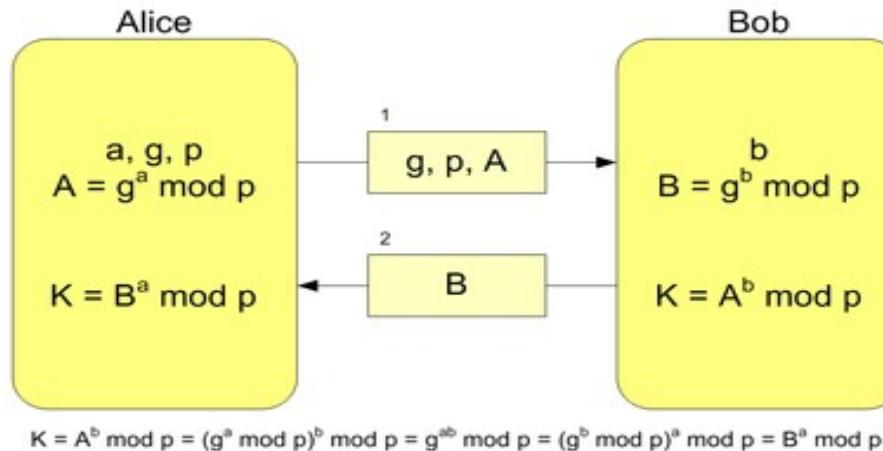
Source: wikipedia

Cet algorithme permet d'établir entre le client et le serveur une clé de chiffrement symétrique (*clé de session*) en échangeant seulement des valeurs publiques.



# La cryptographie

- **Algorithme Diffie-Hellman: Principe**



- Alice choisit un nombre aléatoire  **$a$**  et Bob choisit un nombre aléatoire  **$b$** .
- Alice et Bob s'entendent sur 2 autres nombres  **$g$**  et  **$p$**   
(ils peuvent circuler en clair sur le réseau car il sera difficile d'inverser les exponentiations suivantes).
- Alice calcule  $A = g^a \text{ mod } p$  et Bob calcule  $B = g^b \text{ mod } p$ .
- $A$  est envoyé à Bob et  $B$  est envoyé à Alice.
- Alice calcule  $B^a \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p = (g^{ba} \text{ mod } p) = K$
- Bob calcule  $A^b \text{ mod } p = (g^a \text{ mod } p)^b \text{ mod } p = (g^{ab} \text{ mod } p) = K$

Source: wikipedia

**K = la clé secrète**



# La cryptographie

- **Algorithme Diffie-Hellman: Exemple**

	CLIENT	SERVEUR
Chaque partie génère un nombre aléatoire.	$a=2$	$b=3$
Les parties se mettent d'accord sur 2 nombres communs.	$g=3 \ p=7$ – <i>Echange en clair</i> –	$g=3 \ p=7$
Chaque partie calcule $g^{nbalea} \text{ mod } p$	$\begin{aligned} & g^a \text{ mod } p \\ & = 3^2 \text{ mod } 7 \\ & = 2 \end{aligned}$	$\begin{aligned} & g^b \text{ mod } p \\ & = 3^3 \text{ mod } 7 \\ & = 6 \end{aligned}$
Echange du résultat	6	– <i>Echange en clair</i> –
Chaque partie calcule la <b>clé de session</b> : NbReçu <sup>nbalea</sup> mod p	$\begin{aligned} & 6^2 \text{ mod } 7 \\ & = 1 \end{aligned}$	$\begin{aligned} & 2^3 \text{ mod } 7 \\ & = 1 \end{aligned}$
	$\begin{aligned} & = 6^a \text{ mod } p \\ & = (g^b \text{ mod } p)^a \text{ mod } p \\ & = g^{ba} \text{ mod } p \end{aligned}$	$\begin{aligned} & = 2^b \text{ mod } p \\ & = (g^a \text{ mod } p)^b \text{ mod } p \\ & = g^{ab} \text{ mod } p \end{aligned}$



# La cryptographie

- **Facteurs de fiabilité des techniques de chiffrement**
  - La qualité mathématique de l'algorithme plutôt que le secret de celui-ci.
  - La qualité de l'implémentation de l'algorithme.

En effet, la plupart des attaques visent l'implémentation que l'algorithme lui-même ("buffer overflow").
  - La bonne gestion du secret des clés de chiffrement.

L'algorithme étant la plupart du temps public, si la ou les clés sont connues, il n'y a plus aucune sécurité.



# Qu'est-ce que OpenSSH ?

- OpenSSH repose sur une bibliothèque de fonctions (OpenSSL) qui contient tous les éléments cryptographiques nécessaires au fonctionnement d'OpenSSH.
- C'est la version OpenSource (développée par OpenBSD sous licence BSD) de la suite logicielle proposée par la société SSH Communication Security.
- C'est une application (client/serveur) permettant d'accéder à une machine distante de façon sécurisée.
- C'est aussi le nom du protocole réseau associé à cette application.
- Il remplace notamment les applicatifs *telnet* et *rsh* qui sont considérés comme non-sécurisés.



# Utilités et avantages

- Obtenir un accès distant en garantissant trois des quatre objectifs légaux à atteindre lors de l'échange de données :
  1. Authentification: Seules les personnes autorisées ont accès aux ressources.
  2. Intégrité: On est sûr que le message n'a pas été modifié.
  3. Confidentialité: On est sûr que le message n'a pas pu être lu en clair.

Rem: La 'non répudiation' qui garantit qu'une transaction ne peut être niée par aucune des 2 parties n'est pas assurée par ssh car l'obtention de la clé publique n'est pas toujours sûre... (voir plus loin).

```
# slogin -l pgouwyjl natrezo.bacisat.be ou slogin pgouwyjl@natrezo.bacisat.be
```

- Exécuter une commande à distance de façon sécurisée.  

```
# ssh pgouwyjl@natrezo.bacisat.be cat /etc/shadow
```
- Faire du transfert de fichiers sécurisé (scp, secpelan, winscp ou sftp)
- Encapsuler un autre protocole (tunneling)
- Faire du X11 forwarding (prendre en main le bureau graphique d'une machine distante par ssh -> encapsulation du protocole X dans un tunnel ssh).



# Etablissement d'une communication

## **Etape 1: Négociation entre le client et le serveur**

Phase de négociation entre les 2 parties afin qu'ils s'entendent sur le protocole à employer (ssh1/ssh2) et les cryptosystèmes à utiliser ultérieurement (rsa/dsa/aes/sha...).

Le serveur dispose d'un trousseau de clés (privée/publique). Ces 2 clés sont différentes mais mathématiquement identiques.

Toute information chiffrée avec la clé publique du trousseau ne pourra être déchiffrée qu'avec la clé privée du même trousseau.

Une clé publique peut être dévoilée. Une clé privée doit rester secrète.

Lors de cette étape, le serveur envoie sa clé publique au client



# Etablissement d'une communication

## **Etape 1: Remarque**

Lors de l'acceptation du téléchargement automatique de la clé publique (étape 1), le client accepte, peut-être à tort, que la machine la proposant soit le bon serveur...

- ▶ Transmettre 'manuellement' la clé publique... ou vérifier le *fingerprint* (voir plus loin)



# Etablissement d'une communication

## **Etape 2: Authentification du serveur**

Le client envoie un défi chiffré avec la clé publique du serveur.

Celui-ci déchiffre ce défi avec sa clé privée et renvoie la réponse au client.

Si cette réponse est correcte, le client est certain d'être en relation avec le bon serveur.

L'envoi du défi est chiffré à l'aide d'un cryptosystème dit "asymétrique" (algorithmes rsa ou dsa).

Le client chiffre celui-ci par la clé publique reçue du serveur (étape 1) et le serveur le déchiffre par la clé privée correspondante.

Sécurité:      Règle d'authentification ✓



# Etablissement d'une communication

## **Etape 3: Etablissement d'un canal sécurisé**

Le client et le serveur s'échange de manière sécurisée une clé dite "clé de session" (algorithme de Diffie-Hellman).

A partir de maintenant, un canal sécurisé est créé c'est-à-dire que toutes les données qui passeront par celui-ci seront chiffrées par un algorithme de chiffrement dit "symétrique" (ex. aes) qui utilise la clé de session.

Elles seront également accompagnées d'une empreinte (ex. signature sha).

Sécurité:      Règle de confidentialité (aes) ✓  
                        Règle d'intégrité (sha) ✓



# Etablissement d'une communication

## **Etape 4: Authentification du client**

Méthode traditionnelle: par mot de passe

Méthode plus souple: par clé

## **Etape 5: Communication**

La communication sécurisée peut alors débuter entre le client et le serveur.



# Etablissement d'une communication

## **Authentification du client : Par mot de passe**

Le client envoie un nom d'utilisateur et un mot de passe au serveur par le canal sécurisé.

Le serveur vérifie si l'utilisateur concerné a accès à la machine et si le mot de passe fourni est valide.



# Etablissement d'une communication

## **Authentification du client : Par clés**

Le client doit générer son propre trousseau de clés.

Le client dépose sa clé publique sur le serveur dans sa home directory.

Le serveur va créer un challenge et donner un accès au client si ce dernier parvient à déchiffrer le challenge avec sa clé privée.



# Etablissement d'une communication

## Avantages d'une authentification par clés

Connexion et communication sécurisée sans mot de passe.

Décorellation des mots de passe SSH et Linux.

Pas besoin de reconfigurer le mot de passe Linux si on se fait dérober sa clé privée ...)

Le login de connexion dépend de l'endroit où sera déposée la clé publique du client. Ainsi, déposer ma clé publique dans la home directory de l'utilisateur 'root' de plusieurs serveurs me permettrait d'être 'root' sur ces serveurs sans entrer de mot de passe.

Très utile lors de sauvegardes sécurisées, autonomes et automatisées par scripts.



# Etablissement d'une communication

- **Atelier: Espionner une conversation ssh**

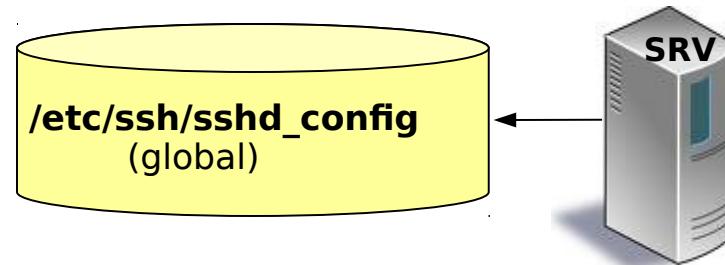
- Lancer *Wireshark* sur le poste client.
- Définir un filtre de capture pour ssh (*tcp.port==22*) et pour l'interface concernée.
- Lancer la capture.
- Etablir une connexion ssh.
- Constatations : Négociation des algorithmes, Diffie-Hellman, chiffrement des paquets ...



# Configuration du serveur



**slogin**  
**ssh**  
**scp**  
**ssh-keygen**  
**ssh-copy-id**  
**[sftp]**  
**[ssh-agent]**  
**[ssh-add]**  
**[ssh-keyscan]**



**sshd** (*serveur ssh - port d'écoute: 22*)

*man sshd ...*

Démarrage / redémarrage / arrêt (rhel5/6.x):

```
# /etc/init.d/sshd start ou restart ou stop  
ou  
# service sshd start ou restart ou stop
```



# Configuration du serveur

## Le fichier sshd\_config: exemple

Port 22 → *Port d'écoute de sshd*  
Protocol 2 → *Protocoles à supporter (ssh2)*  
ListenAddress 0.0.0.0 → *Adresse Ip de l'interface d'écoute  
(ici toutes les interfaces)*

HostKey /etc/ssh/ssh\_host\_key      HostKey /etc/ssh/ssh\_host\_rsa\_key      HostKey /etc/ssh/ssh\_host\_dsa\_key }      *Toutes les clés privées du serveur*

Key\_regeneration\_interval 3600      ServerKeyBits 1024 }      *Regénération d'une clé de session de 1024  
bits après 3600 sec. de connexion*

LoginGraceTime 120 → *Temps accordé à la procédure de login*



# Configuration du serveur

**RSAAuthentication yes** →

*Authentification par paire de clés SSH1 acceptée.*

**PubkeyAuthentication yes** →

*Authentification par paire de clés SSH2 rsa ou dsa acceptée.*

**HostbasedAuthentication no**  
**IgnoreRhosts yes**  
**RhostsRSAAuthentication no**

→ *Pour empêcher les authentifications de type remote (car non-sécurisées).*

**PasswordAuthentication yes** →

*Authentification par mot de passe (rabattage possible en cas d'échec à l'authentification par clés).*

**PermitEmptyPasswords no** →

*Mais pas pour les comptes sans mdp.*

**KeepAlive yes** → *Pour éviter que la connexion reste ouverte si le client disparaît, le serveur coupe la connexion s'il ne reçoit plus du client un message « Je suis en vie » envoyé régulièrement par celui-ci.*



# Configuration du serveur

## Le fichier sshd\_config: Autres directives

```
DenyUsers
AllowUsers test admin }
```

*Autoriser les deux utilisateurs (test et admin) et aucun autre à se connecter.*

*Voir aussi les directives  
AllowGroups/DenyGroups*

*PermitRootLogin yes —— Le root peut-il se connecter ?*

*PermitRootLogin without-password*

*Le root ne peut se connecter que par paire de clés. Cela évite les tentatives d'attaque ssh par force brute sur le compte root.*



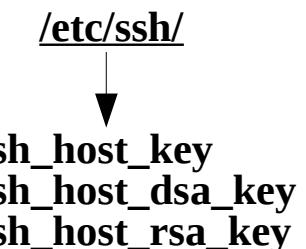
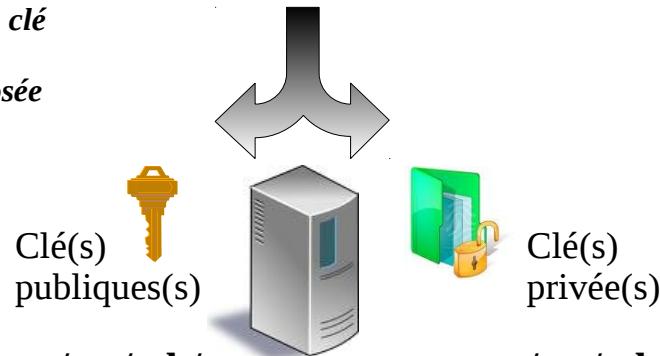
# Implémentation par mot de passe

*Si le client ne possède pas encore la clé publique du serveur, une invite de téléchargement de celle-ci est proposée lors de la première connexion...*



`~/.ssh/known_hosts`

*Trousseau de clés généré lors de l'installation du service sshd.*



*Rem. ssh-keygen -t rsa -b 1024 → pour générer une clé de 1024 bits ...*



# Implémentation par mot de passe

## Le fingerprint

Lors du téléchargement de la clé publique, le client devrait s'assurer qu'il s'agit bien de celle du serveur ciblé au risque de se connecter à un pirate qui aurait usurpé l'IP du vrai serveur.

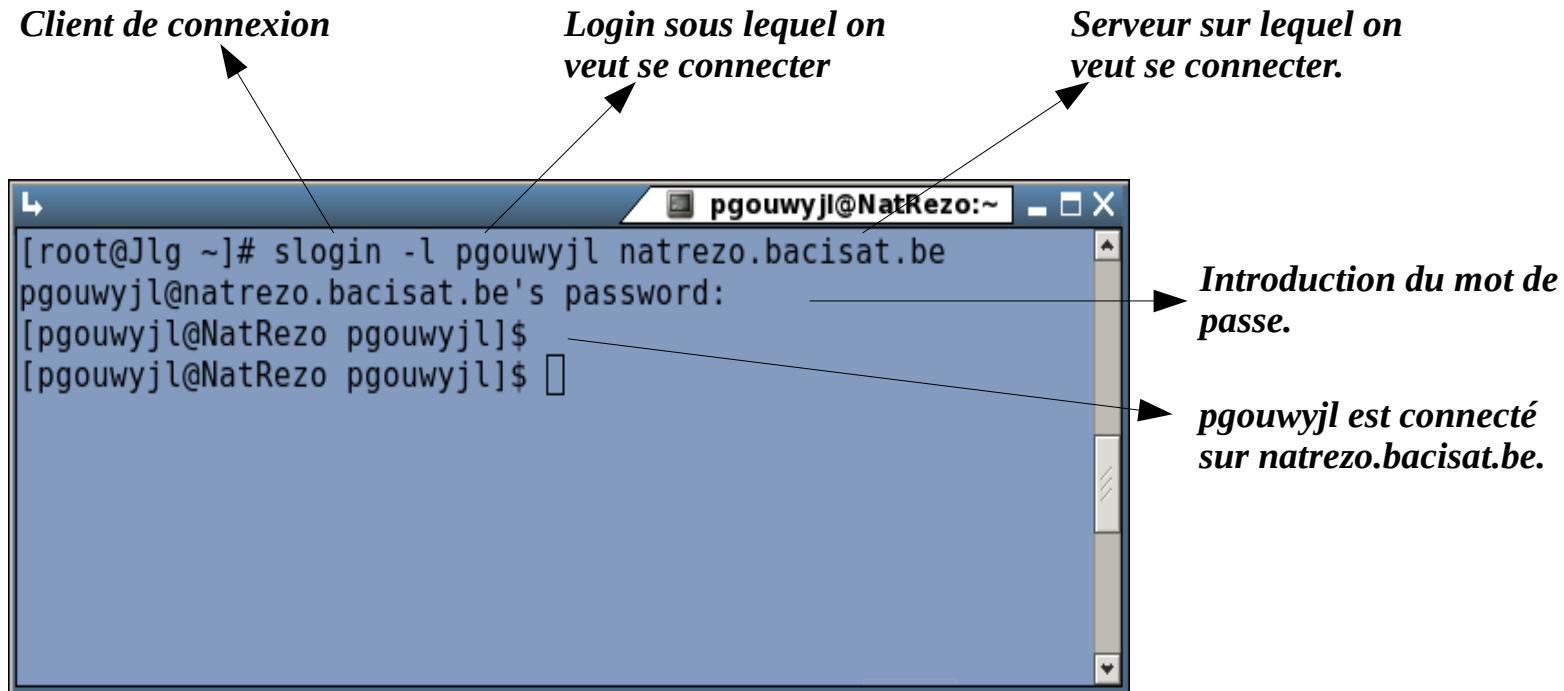
### Une bonne méthode:

- L'administrateur du serveur génère un 'fingerprint' (chaîne générée lors de la génération de la clé publique du serveur).  
La commande 'ssh-keygen -lf /etc/ssh/ssh\_host\_rsa\_key.pub' permet d'afficher cette empreinte.  
Ex. 2048 fe:91:17:91:c1:bd:ab:ae:5e:05:8b:70:40:1b:e8:c2 (**RSA**)
- Le client demande le 'fingerprint' à l'administrateur.
- Le client compare le 'fingerprint' présenté lors du téléchargement à celui reçu de l'administrateur.
  - ✓ Cette procédure ne garantit en rien la non-réputation car elle n'est pas obligatoire et systématique.



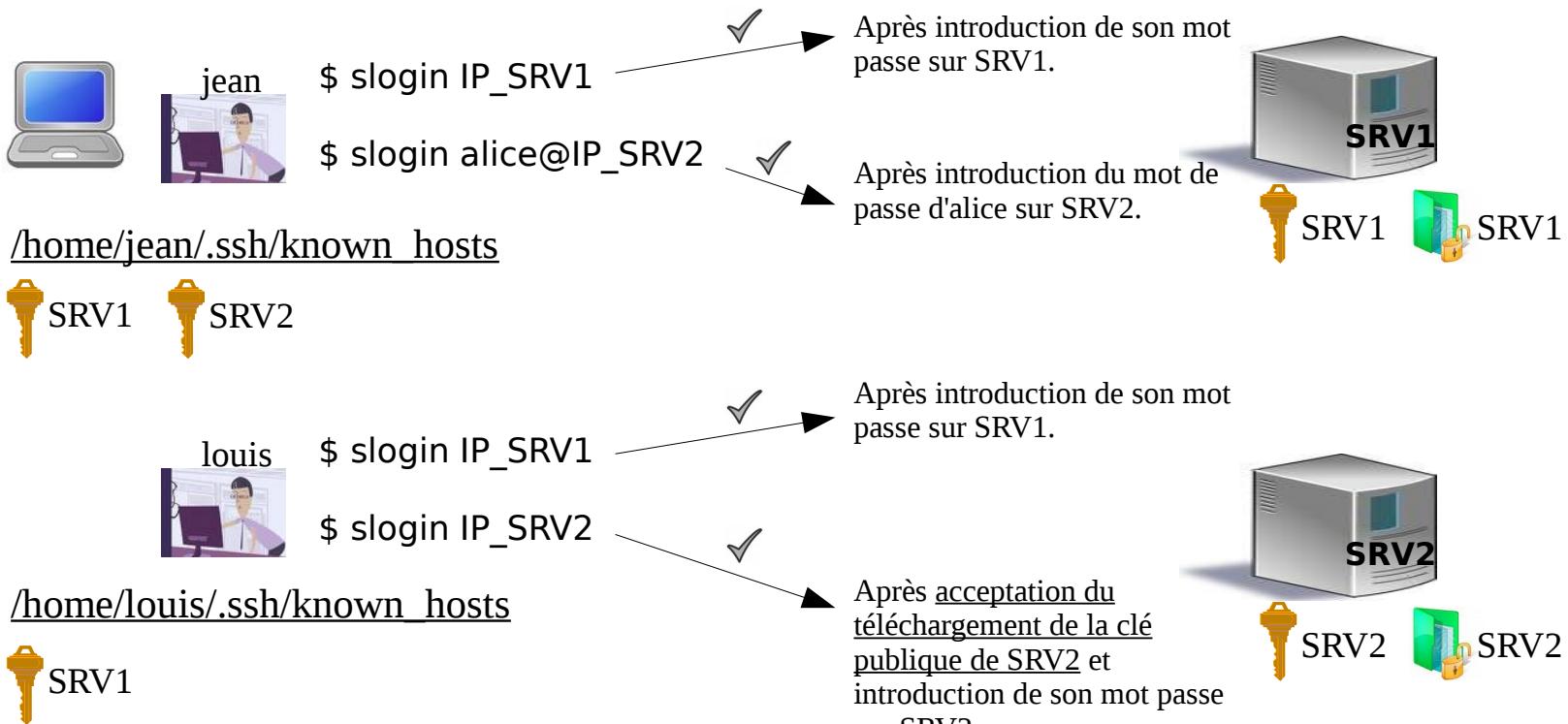
# Implémentation par mot de passe

## Exemple



# Implémentation par mot de passe

On peut trouver un fichier **known\_hosts** dans plusieurs home directories.  
Il peut contenir plusieurs clés publiques appartenant à des serveurs différents.

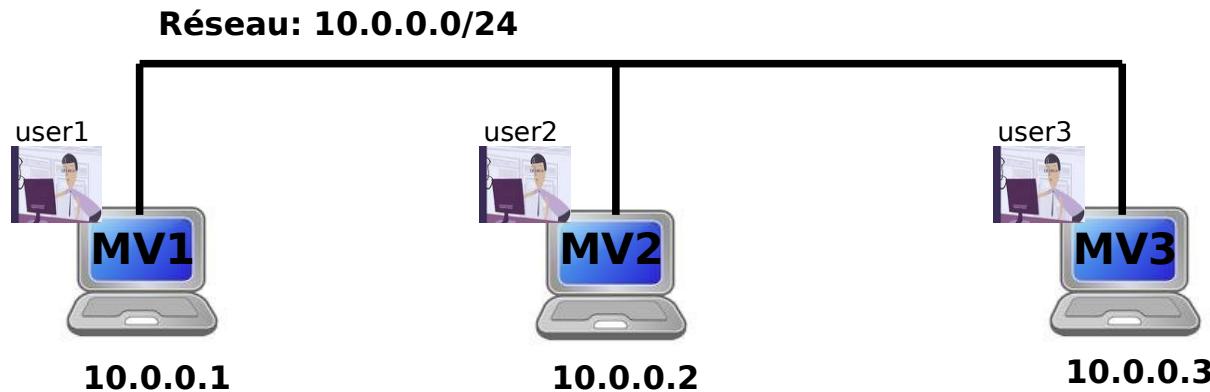


# Remarques

- Attention aux permissions sur les fichiers et répertoires (voir labo.)
- Fichiers logs: */var/log/messages* & */var/log/secure*
- */usr/sbin/sshd -d* pour lancer le serveur en mode debug...
- Le protocole SSH1 souffre de failles de sécurité dans l'intégrité des données envoyées ou reçues, le rendant vulnérable à des attaques actives.
- Mieux vaut configurer le serveur pour n'accepter que les connexions sous protocole SSH2.
- SFTP n'est opérationnel que sous le protocole SSH2.



# Exercice: Connexion par mot de passe



Chaque machine doit accepter des connexions par mot de passe.

Pour ce faire:

- vérifiez qu'aucun coupe-feu ne tourne sur aucune machine
- vérifiez sur chaque machine qu'un trousseau de clés existe



# Exercice: Connexion par mot de passe (suite)

- configurez un service sshd sur chaque machine et lancez le service (bien vérifiez que la directive PasswordAuthentication est à yes)
- créez user1 sur MV1, user2 sur MV2 et user3 sur MV3
- testez les connexions suivantes:

```
user1@MV1$ slogin user2@10.0.0.2
user1@MV1$ slogin user3@10.0.0.3
```

```
user2@MV2$ slogin user1@10.0.0.1
user2@MV2$ slogin user3@10.0.0.3
```

```
user3@MV3$ slogin user2@10.0.0.2
user3@MV3$ slogin user1@10.0.0.1
```



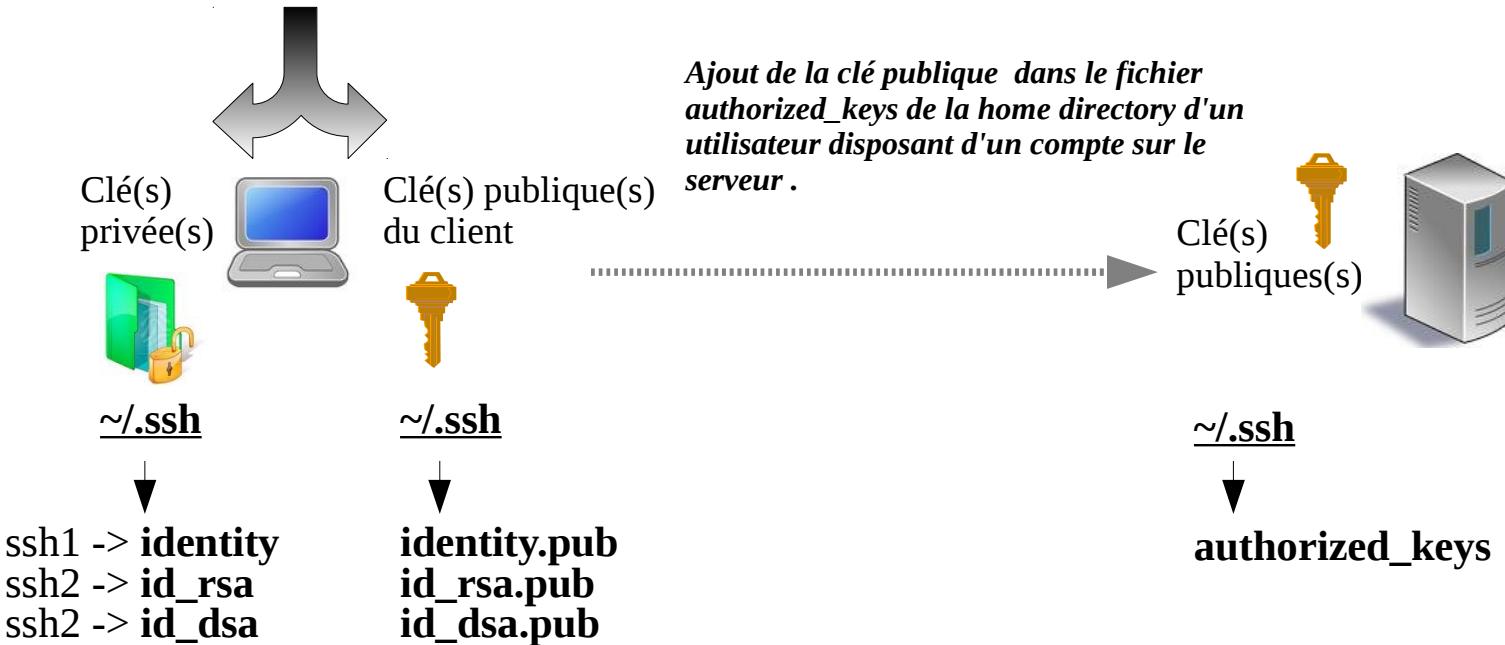
# Exercice: Connexion par mot de passe (suite)

- visualisez les ports d'écoute (client/serveur) à travers une connexion
- utilisez *ssh* et *scp* (après avoir consulter les pages de manuel ☺)
- explorez les fichiers `/etc/ssh/*` et `~/.ssh/known_hosts`



# Implémentation par clés

```
# ssh-keygen -t rsa1  
# ssh-keygen -t rsa  
# ssh-keygen -t dsa
```



# Implémentation par clés

## Exemple



# Implémentation par clés

On peut trouver un fichier **authorized\_keys** dans plusieurs home directories.  
Il peut contenir plusieurs clés publiques appartenant à des clients différents.



✓ \$ slogin alice@IP\_SRV1  
✓ \$ slogin arthur@IP\_SRV2



✓ \$ slogin bob@IP\_SRV1



✓ \$ slogin bob@IP\_SRV1



/home/alice/.ssh/authorized\_keys



/home/bob/.ssh/authorized\_keys



/home/arthur/.ssh/authorized\_keys

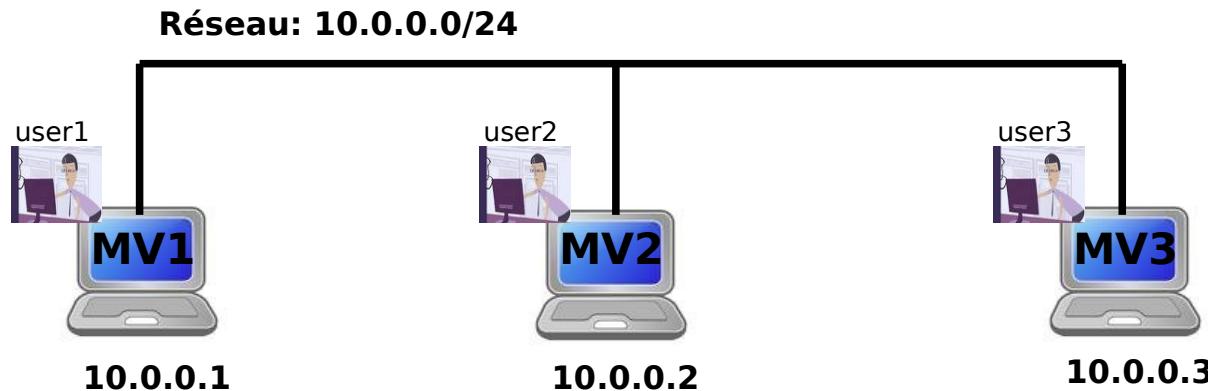


# Remarques

- Si une autre personne a accès à votre compte (ou vole votre disque dur), il pourrait copier votre clé privée et s'en servir pour accéder à votre compte sur un serveur SSH sauf si elle a été protégée par une PASSPHRASE (cryptée via l'algorithme des) lors de sa génération ...
- L'agent ssh: Outil qui charge en mémoire la clé privée pour éviter d'avoir à rentrer la PASSPHRASE plusieurs fois ...



# Exercice : Connexion par clés



Faites en sorte que *user1* puisse se connecter sur MV3 sous *user3* sans entrer de mot de passe. Pour ce faire:

- root@MV3# → vérifie des clauses RSAAuthentication et PubkeyAuthentication à yes dans /etc/ssh/sshd\_config
- user1@MV1\$ → génère un trousseau de clés de type RSA (sans passphrase)
- user3@MV3\$ → crée un dossier ~/.ssh



# Exercice : Connexion par clés (suite)

- user1@MV1\$ → copie sa clé publique dans un fichier authorized\_keys de user3 (attention aux permissions)
  
- user1@MV1\$ → se connecte sous user3 sur MV3.

Explorez les fichiers ~/.ssh/\* et ~/.ssh/authorized\_keys



# Utilités et avantages

## **Avantages fonctionnels du Secure Shell:**

Pouvoir travailler de manière sécurisée sur une machine sans être obligé de se déplacer.

Pouvoir facilement transférer des fichiers de manière sécurisée.

Pouvoir facilement centraliser et automatiser l'administration d'un ensemble de machines (scripts).



# Utilités et avantages

## Avantages techniques du Secure Shell:

### Authentification forte:

- si connexion par paire de clés (pas de circulation de mot de passe)
- protection supplémentaire: possibilité de protéger sa clé privée par une passphrase.
- algorithme de chiffrement asymétrique très puissant (RSA/DSA)
- possibilité de vérifier la validité de la clé publique du serveur lors de son téléchargement via son empreinte (fingerprint)

### Alerte contre les attaques de type 'man-in-the-middle':

- si lors d'une connexion, le message REMOTE HOST IDENTIFICATION HAS CHANGED apparaît chez le client, cela signifie que la clé publique du serveur n'est plus la même que celle reçue lors de la 1ère connexion...
  - l'administrateur a regénéré un nouveau trousseau de clés
  - ou
  - un pirate tente de détourner votre connexion



# Gestion d'un parc Linux

- Mise à jour logicielle.
- Changement de configuration.
- Supervision des stations.
- ...

Bien souvent, les mêmes manipulations à répéter sur des dizaines de stations 😞.

☞ Besoin d'automatiser ces tâches.

Une solution: ssh



# Gestion d'un parc Linux

## Solution 1:

Intervenir localement et effectuer les interventions à la main sur chaque station.



Nécessité de se déplacer.

Risque d'erreur.

Tâche trop lourde si le parc est important et décentralisé.



# Gestion d'un parc Linux

## Solution 2:

Se connecter via ssh (par mot de passe) sur chaque station et effectuer les interventions à distance et manuellement sur chaque station.



Risque d'erreur.

Tâche trop lourde si le parc est important.  
(1000 stations = 1000 connexions + 1000 interventions)



# Gestion d'un parc Linux

## Solution 3:

Se connecter via ssh (par mot de passe) sur chaque station et automatiser à distance les interventions (scripts) sur chaque station.



Tâche trop lourde si le parc est important.  
(1000 stations = 1000 connexions)



# Gestion d'un parc Linux

## Solution 4:

Se connecter via ssh (par clé) sur chaque station et automatiser à distance les interventions (scripts) sur chaque station.

### ☞ **SOLUTION RETENUE**



# Implémentation de la solution

Sur la station d'administration

- Un trousseau de clés doit être généré.
- Un fichier `/root/admin/ip.txt` contiendra les @ip de toutes les stations à administrer.
- Un dossier `/root/admin/logs` contiendra les logs générés par les scripts d'administration.
- Un dossier `/root/admin/sh` contiendra les scripts d'administration.

Sur les stations du parc

- La clé publique de la station d'administration doit être recopiée dans la home directory de root de chaque station (fichier `authorized_keys`).
- sshd doit tourner.



# Exercices: Gestion d'un parc Linux

## Exercice 1:

Ecrivez et testez un script (`pushkey.sh`) qui déploie la clé publique de root de la machine d'administration (MV1) vers toutes les stations du parc.

## Exercice 2:

Ecrivez et testez un script (`haltall.sh`) qui éteint toutes les stations du parc encore « on-line ».

Programmez l'exécution de ce script à 21h00 tous les jours.  
Pour ce faire le package crontab doit être installé...



# Exercices: Gestion d'un parc Linux

## Exercice 3:

Ecrivez et testez un script (`chpwdroot.sh`) qui change mot de passe de root sur toutes les stations du parc. Le nouveau de passe est passé en argument au script.

## Exercice 4:

Ecrivez et testez un script (`alladduser.sh`) qui ajoute un compte utilisateur à chaque station du parc.

Le nom de l'utilisateur (ex. `toto`) sera passé en argument. Son mot de passe sera identique à son nom.



# Exercices: Gestion d'un parc Linux

## Exercice 5:

Ecrivez et testez un script (`chgrub.sh`) qui permet de changer le 'timeout' du multi-boot de chaque machine à 5 secondes.

## Exercice 6:

Ecrivez et testez un script (`chnetwork.sh`) qui change la configuration ip de chaque station du parc afin de les disposer sur le réseau d'adresse 192.168.0.0/24.

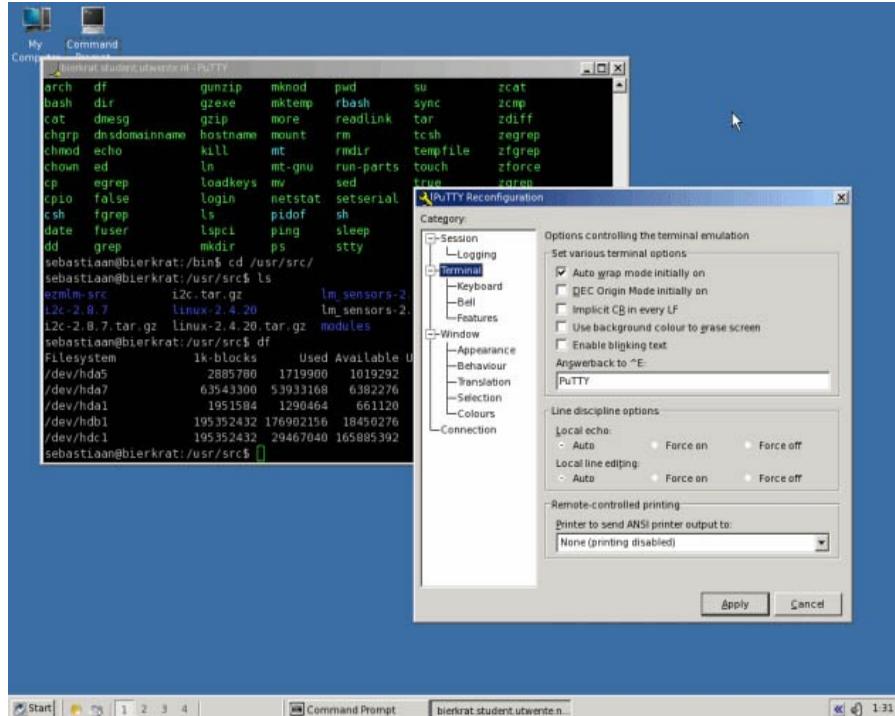


# Quelques outils

## Pour Windows et Linux

**PUTTY:** Utilitaires mode texte pour clients telnet et ssh , copie de fichiers de manière sécurisée, générateur de trousseau de clés...

➤ <http://www.putty.org/>

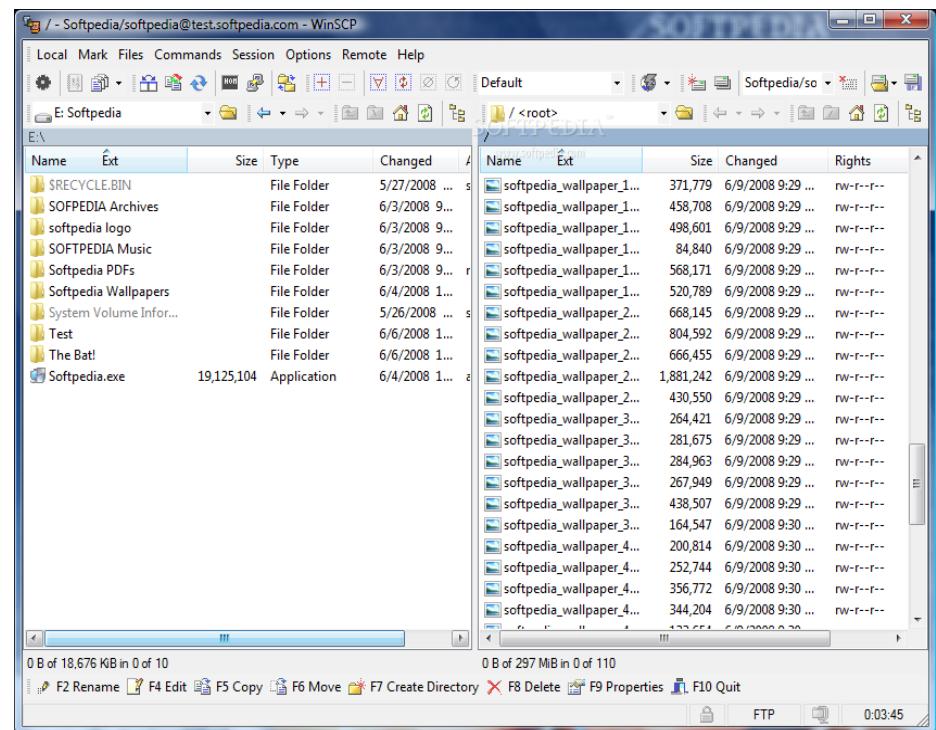
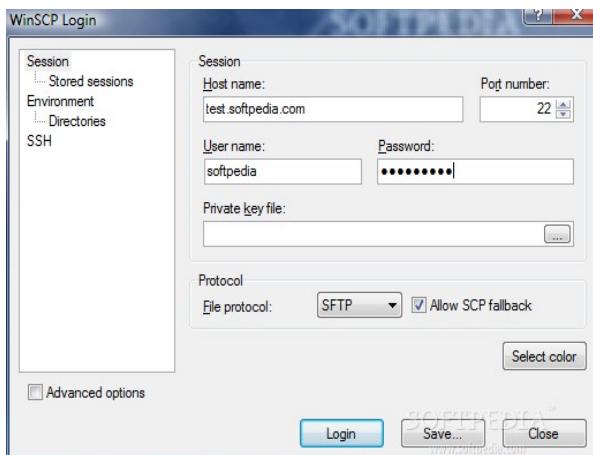


# Quelques outils

## Pour Windows

**WINSCP:** Utilitaire graphique permettant le transfert de fichiers par 'drag and drop'.

☛ <http://winscp.net/eng/docs/lang:fr>

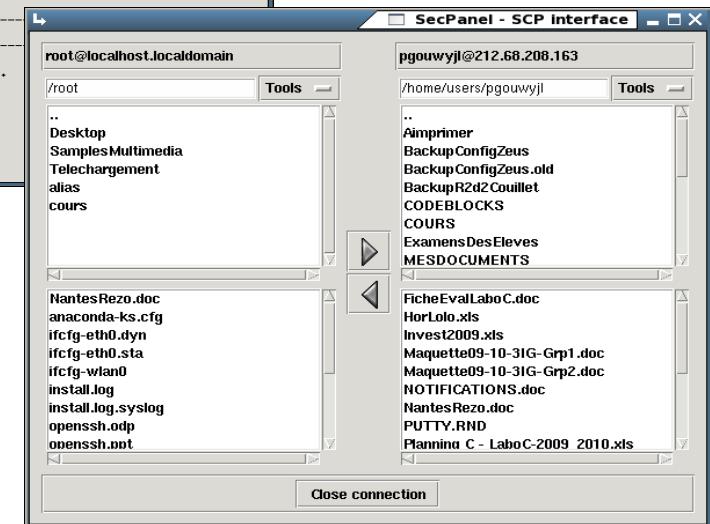
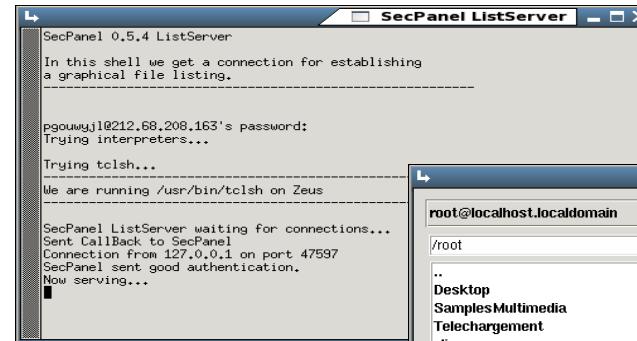
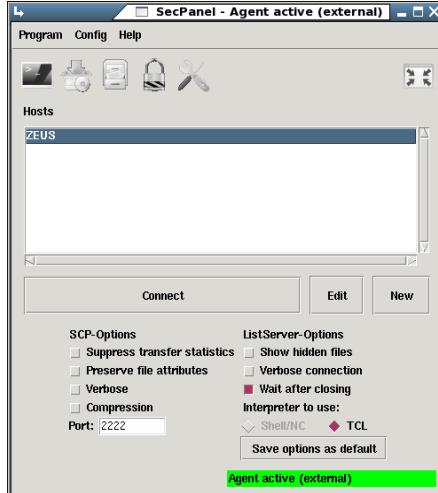


# Quelques outils

## Pour Linux

**secpanel:** Utilitaire graphique (écrit en Tcl/Tk) permettant de combiner les fonctionnalités de PUTTY et de WINSSCP.

➤ <http://themediahost.de/secpanel/>

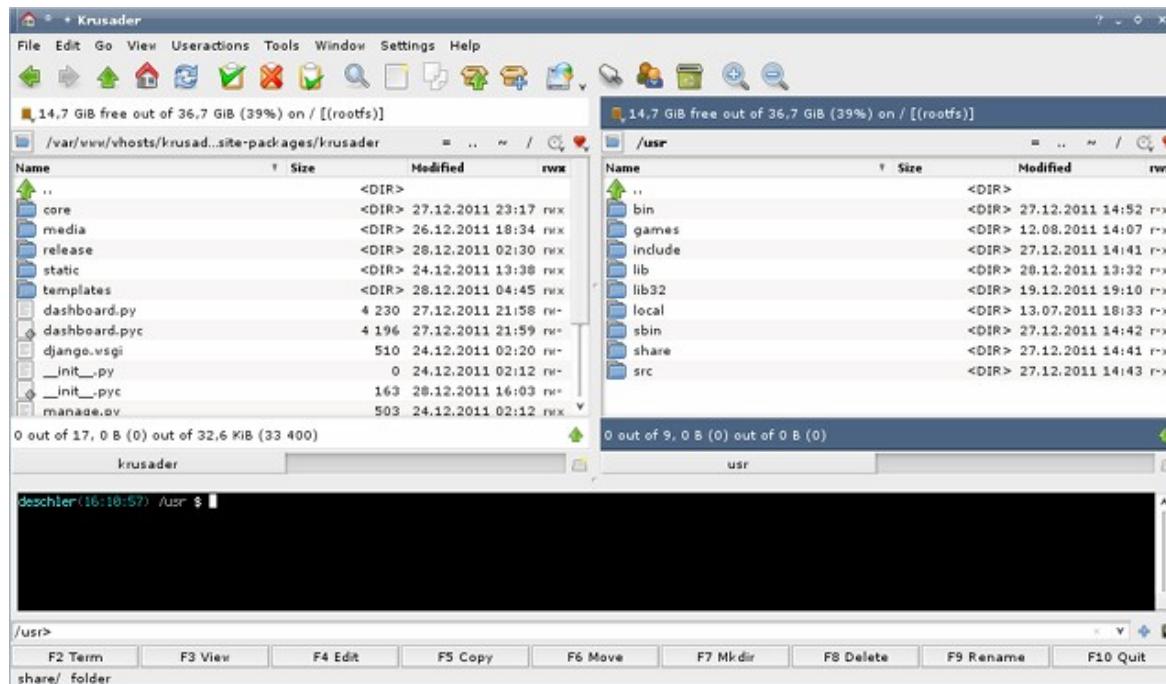


# Quelques outils

## Pour Linux et Windows

**Krusader:** Utilitaire graphique sous KDE permettant le transfert de fichiers par 'drag and drop'.

► <http://www.krusader.org>



# Quelques outils

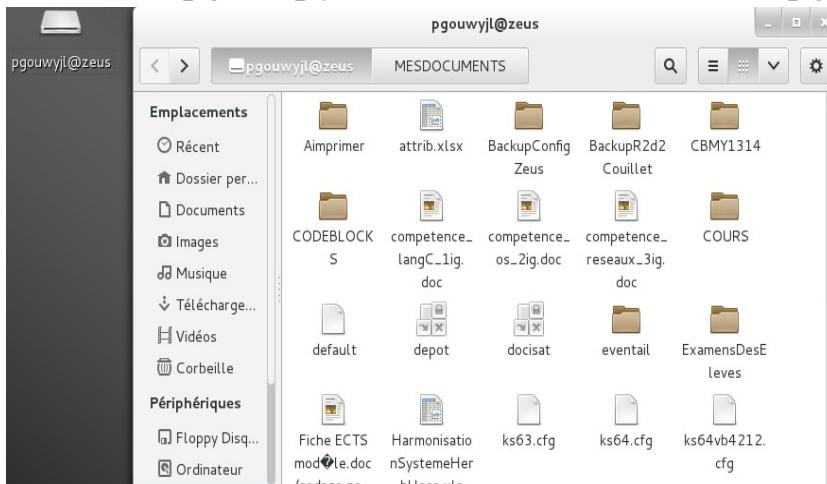
## Pour Linux et Windows

**sshfs:** Sert à monter un fs distant, à travers une connexion SSH, le tout avec des droits utilisateur. L'avantage est de manipuler les données distantes avec n'importe quel gestionnaire de fichier (Nautilus, Konqueror ou même la ligne de commande), ce qui est bien plus pratique que la commande scp couplée avec ssh.

Bonne alternative à certains utilisateurs Windows qui utilisaient WinSCP.

- [http://www.server-world.info/en/note?os=CentOS\\_6&p=fuse](http://www.server-world.info/en/note?os=CentOS_6&p=fuse)  
<http://doc.ubuntu-fr.org/sshfs>

```
$ sshfs pgouwyjl@zeus.moi.be:/home/pgouwyjl /home/my_dir
```

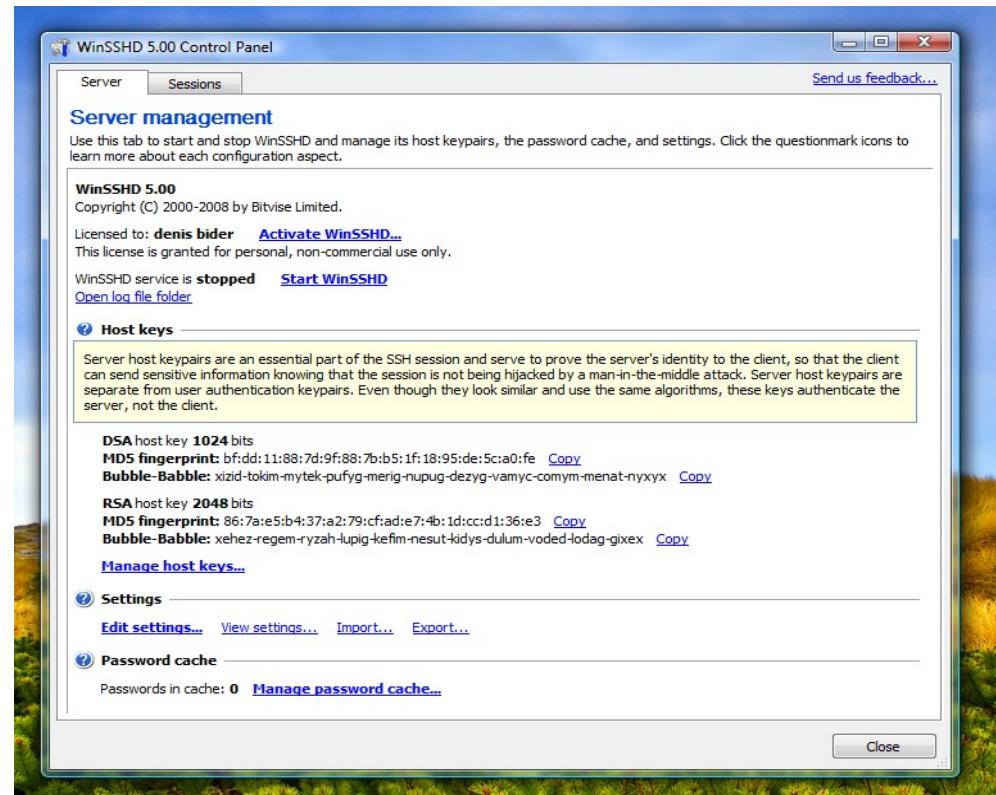


# Quelques outils

## Pour Windows

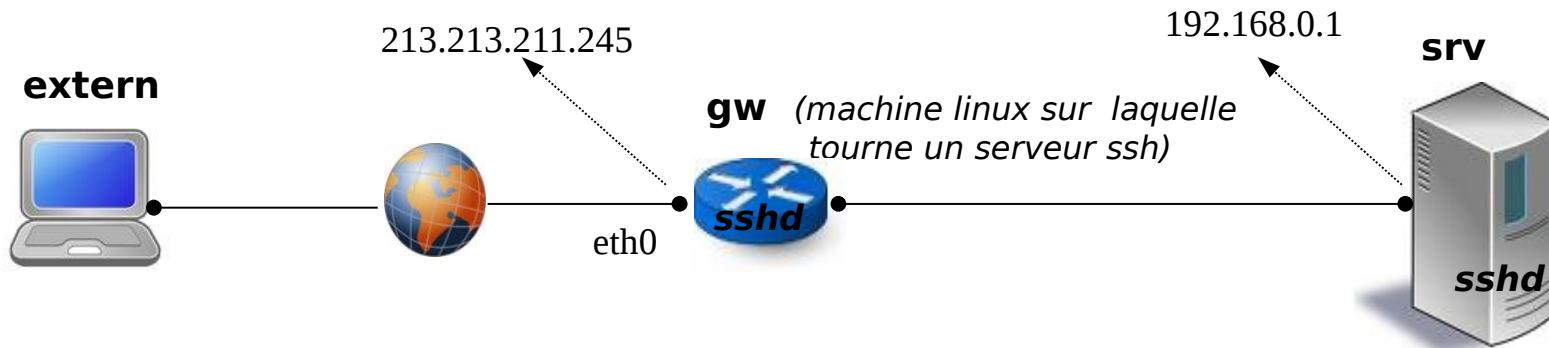
**WinSSHD:** Serveur SSH pour Windows.

➤ <http://www.bitvise.com/winsshd>



# Le port forwarding

- **Maquette**



- **Comment se connecter sur *srv* à partir de *extern* ?**

1° **extern\$** ssh 213.213.211.245  
**gw\$** ssh 192.168.0.1  
**srv\$**



2 connexions et sshd nécessaire sur gw



# Le port forwarding

## 2° Faire du port forwarding

Sur gw, détourner les connexions tcp entrantes par eth0 sur un port spécifique (ex. 2222) vers une autre machine (srv) sur son port ssh (22).

```
gw# iptables -t nat -I PREROUTING -p tcp -i eth0 --dport 2222 -j DNAT --to 192.168.0.1:22
```

```
gw# iptables -L -t nat    (Pour vérifier)
```

```
extern# ssh -p 2222 213.213.211.245  
(Introduire le mot de passe)
```

```
srvlou$ who  
root pts/2 <Ip_Ad_extern>
```



1 connexion et sshd nécessaire uniquement sur srv



# Le port forwarding

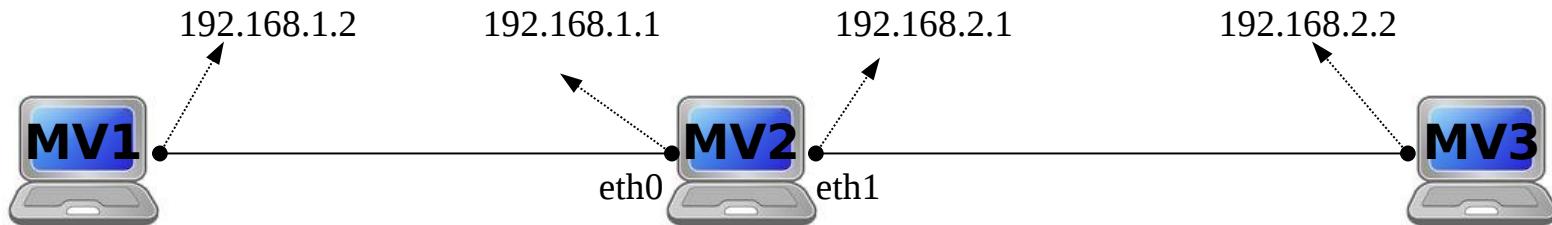
- **Remarques**

- Le port cible (ici 2222) doit être disponible sur la passerelle.
- Commande iptables: voir chapitre 'Le firewalling et Netfilter'
- Le forwarding de port est configurable sur n'importe quel routeur physique actuel.
- N'importe quel port peut être forwardé vers n'importe quel autre port.  
(ex. Le port 8080 de gw vers le port 80 d'une autre machine ou même le port 80 de gw vers le port 80 d'une autre machine ...)
- Avantage: permet d'accéder à des services qui tournent derrière une passerelle sur un réseau privé et protégé par firewall, proxy ...



# Exercice: Le port forwarding

- **Maquette**



- Réalisez la maquette présentée ci-dessus (virtualisation: réseau interne)
- Configurez et testez un forwarding de port de MV1 vers MV3 en passant par MV2

### Remarque

- Le port forwarding implique l'ip forwarding.
- MV2 est donc une passerelle.
- MV1 et MV3 peuvent se toucher mutuellement.
- Ici, le port forwarding est donc inutile...
  - ... utile si MV2 sépare un réseau privé d'un réseau public (nat)
- Le nating sur MV2 doit être activé !



# Le tunneling

- SSH permet de véhiculer n'importe quel flux TCP dans un tunnel crypté d'une session SSH. C'est le petit cousin du VPN (Virtual Private Network)
- Donc, le flux de l'application considérée (très souvent non crypté) sera encapsulé à l'intérieur du "tunnel" SSH. Il ne sera donc plus véhiculé en clair entre les ports client et serveur habituels.
- Le tunneling repose sur 2 grands principes: la redirection (ou relayage) de ports et l'encapsulation de protocoles.
- Applications classiques possibles:
  - Tunneliser sa lecture de mails sur un client POP3.
  - Tunneliser toute une connexion et un flux Ftp.
  - Tunneliser une conversation http.
  - Tunneliser un accès à une base de données.
  - ...
- Si votre firewall (ou proxy) bloque certaines applications mais pas le port 22, celles-ci pourront néanmoins être disponibles à condition d'être véhiculées par un tunnel ssh...



# Le tunneling

Le tunneling peut s'opérer de 2 façons différentes:

- Local (-**L**)

C'est le c**L**ient qui devra créer le tunnel vers le service désiré puis l'utiliser.

- Remote (-**R**)

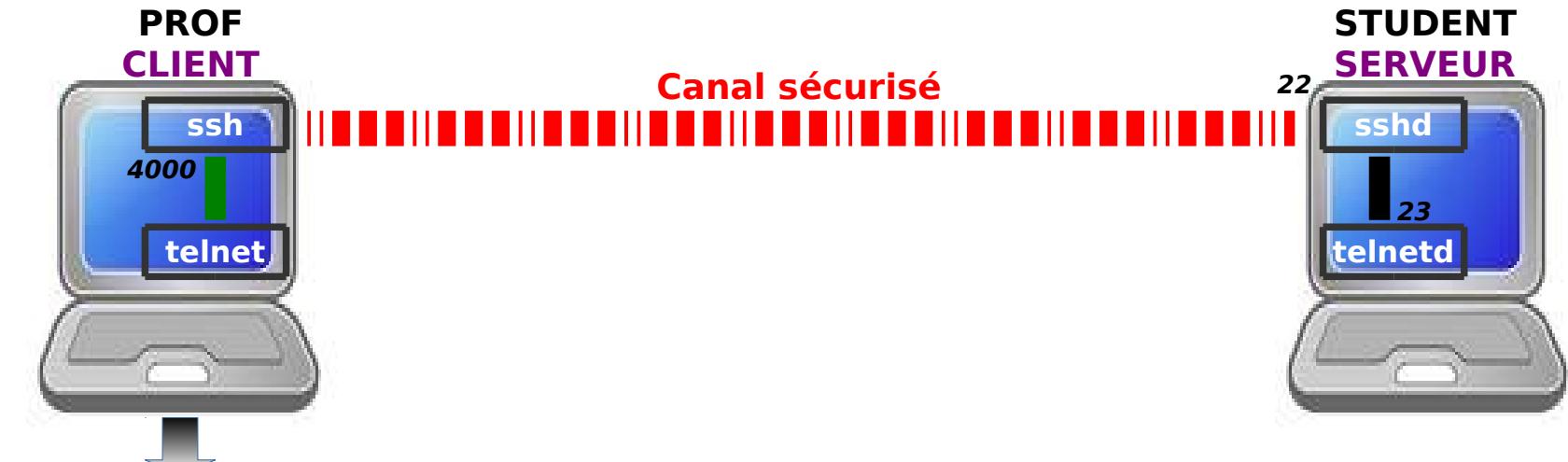
C'est le se**R**veur qui devra créer le tunnel et mettre à la disposition du client un service.



# Le tunneling

Local - Modèle à 2 machines - Soit 'tunneliser' une conversation telnet...

## Connexion telnet de PROF → STUDENT



```
$ ssh -N -f -L 4000:IP_STUDENT:23 user@IP_STUDENT
```

```
$ telnet localhost 4000
```

- ① *Création d'un canal SSH du client SSH vers le serveur SSH ...*
- ② *Tout ce qui arrivera à sshd par ce canal sera redirigé vers le port 23*
- ③ *Tout ce qui entrera par le port 4000 sera envoyé dans le canal sécurisé.*



# Le tunneling

## Local - Modèle à 2 machines - Constatations

- L'option `-N` permet de n'exécuter aucune commande distante.
- L'option `-f` permet de lancer ssh en background.
- Le port local (ici 4000) doit être libre sur la machine concernée.
- Tout ce qui transite par le canal est chiffré donc sécurisé.  
Seules les données transitant en local dans les 2 machines ne sont pas cryptées.
- Pour utiliser le canal, il suffit d'envoyer ses données sur le port 4000 de la machine locale (*telnet localhost 4000*).
- Un modèle à 2 machines implique l'utilisation des 2 mêmes adresses ip dans la commande.
- Le canal reste ouvert même après fermeture de la communication telnet.
- Le client n'a besoin d'aucun daemon serveur sur sa machine mais la création du tunnel et son utilisation est à sa charge. De plus, il a besoin d'un compte sur le serveur.
- Le serveur sshd pourrait être le point de convergence de plusieurs canaux.



# Le tunneling

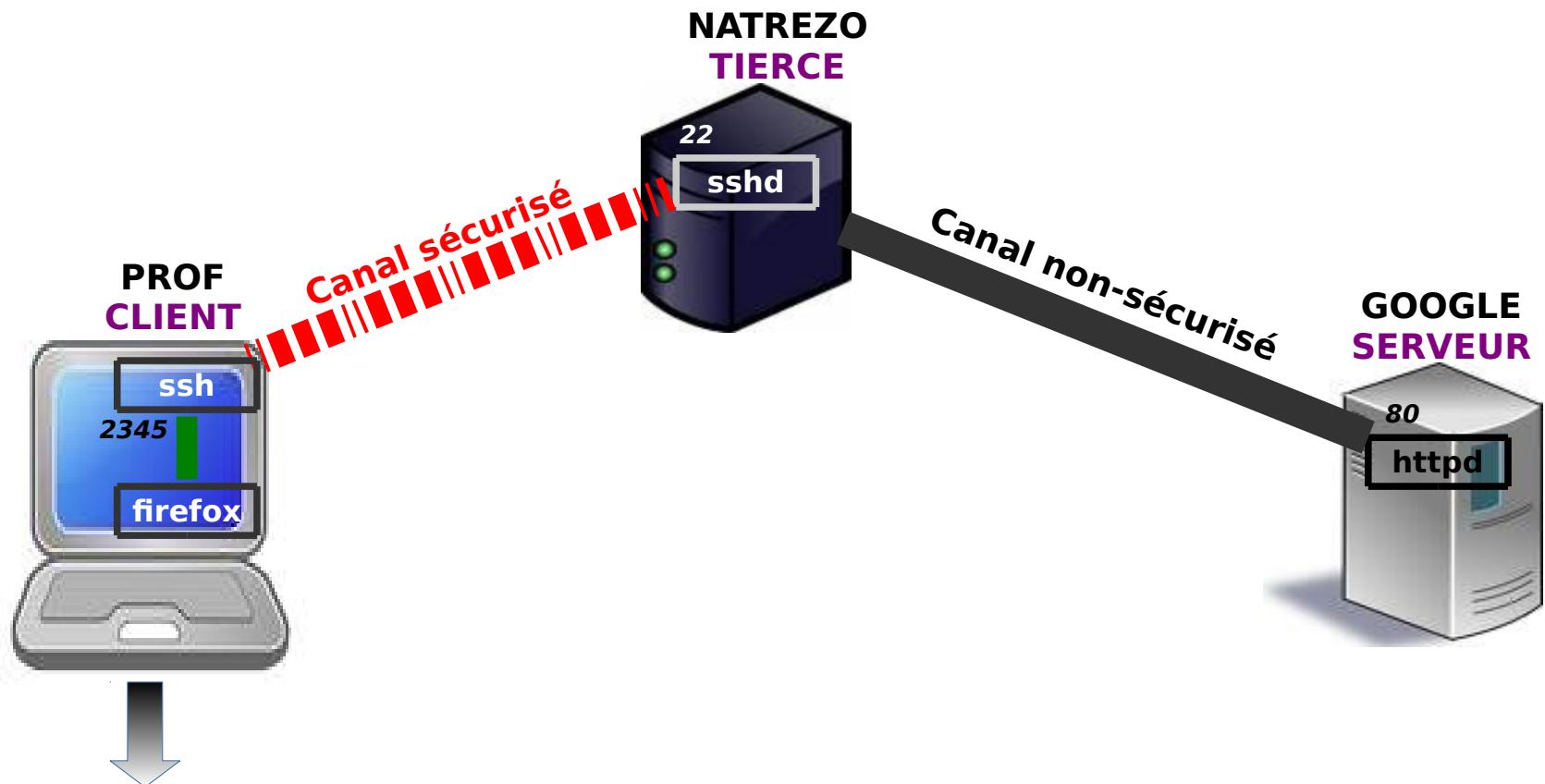
## Local - Modèle à 2 machines - Atelier

- Après création du canal:
  - relevez la présence de celui-ci sur les 2 machines.
  - relevez les ports ouverts sur les 2 machines. Cela correspond-t-il au schéma ?
- Après ouverture de la communication telnet, relevez les ports ouverts sur les 2 machines.  
Cela correspond-t-il au schéma ?
- Fermez le canal sécurisé.



# Le tunneling

Local - Modèle à 3 machines - Soit 'tunneliser' une conversation http...



```
$ ssh -N -f -L 2345:www.google.be:80 user@natrezo.bacisat.be
lynx http://localhost:2345
```



# Le tunneling

## Local - Modèle à 3 machines - Constatations

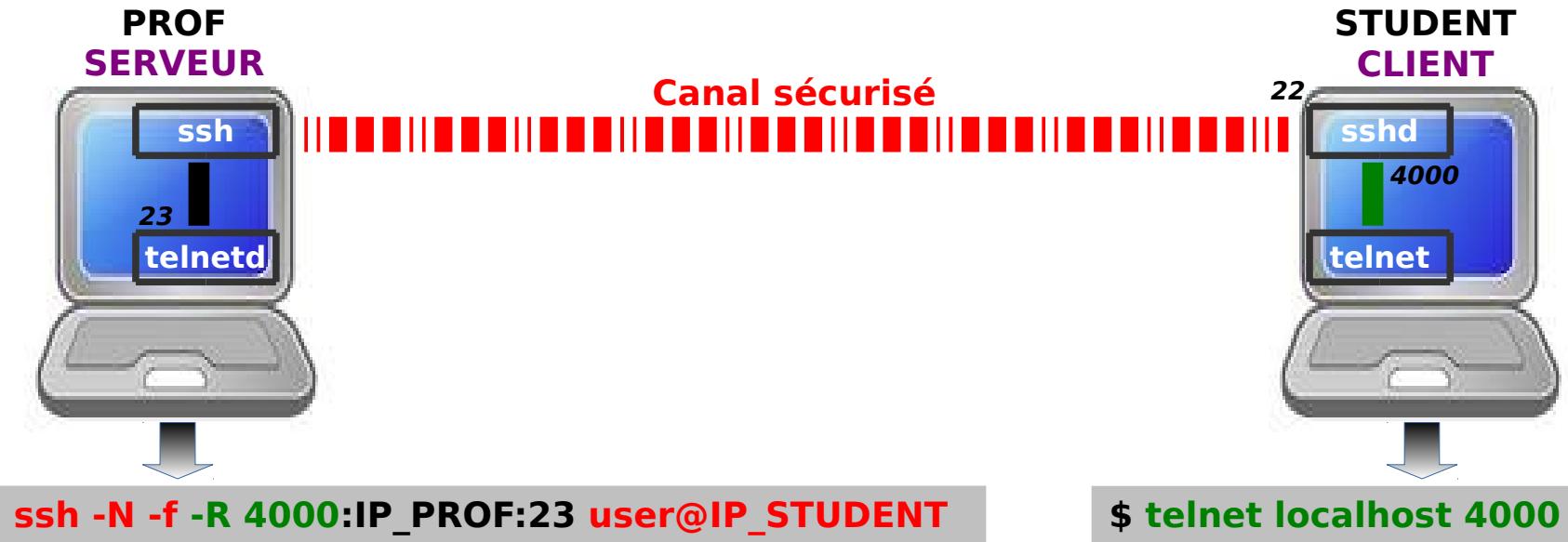
- Un modèle à 3 machines implique l'utilisation des 2 adresses ip différentes dans la commande.
- Tout fonctionne comme si <http://localhost:2345> pointait directement sur [www.google.com:80](http://www.google.com:80)
- Le client n'a besoin d'aucun daemon serveur sur sa machine mais la création du tunnel et son utilisation est à sa charge. De plus, il a besoin d'un compte sur la machine tierce.
- Ici, le sshd et le service à toucher sont sur des machines différentes.
- La communication est chiffrée seulement entre le client et la machine tierce.



# Le tunneling

Remote - Modèle à 2 machines - Soit 'tunneliser' une conversation telnet...

Connexion telnet de STUDENT → PROF



- ① *Création d'un canal SSH du client SSH vers le serveur SSH ...*
- ② *Tout ce qui arrivera à ssh par ce canal sera redirigé vers le port 23*
- ③ *Tout ce qui entrera par le port 4000 sera envoyé dans le canal sécurisé.*



# Le tunneling

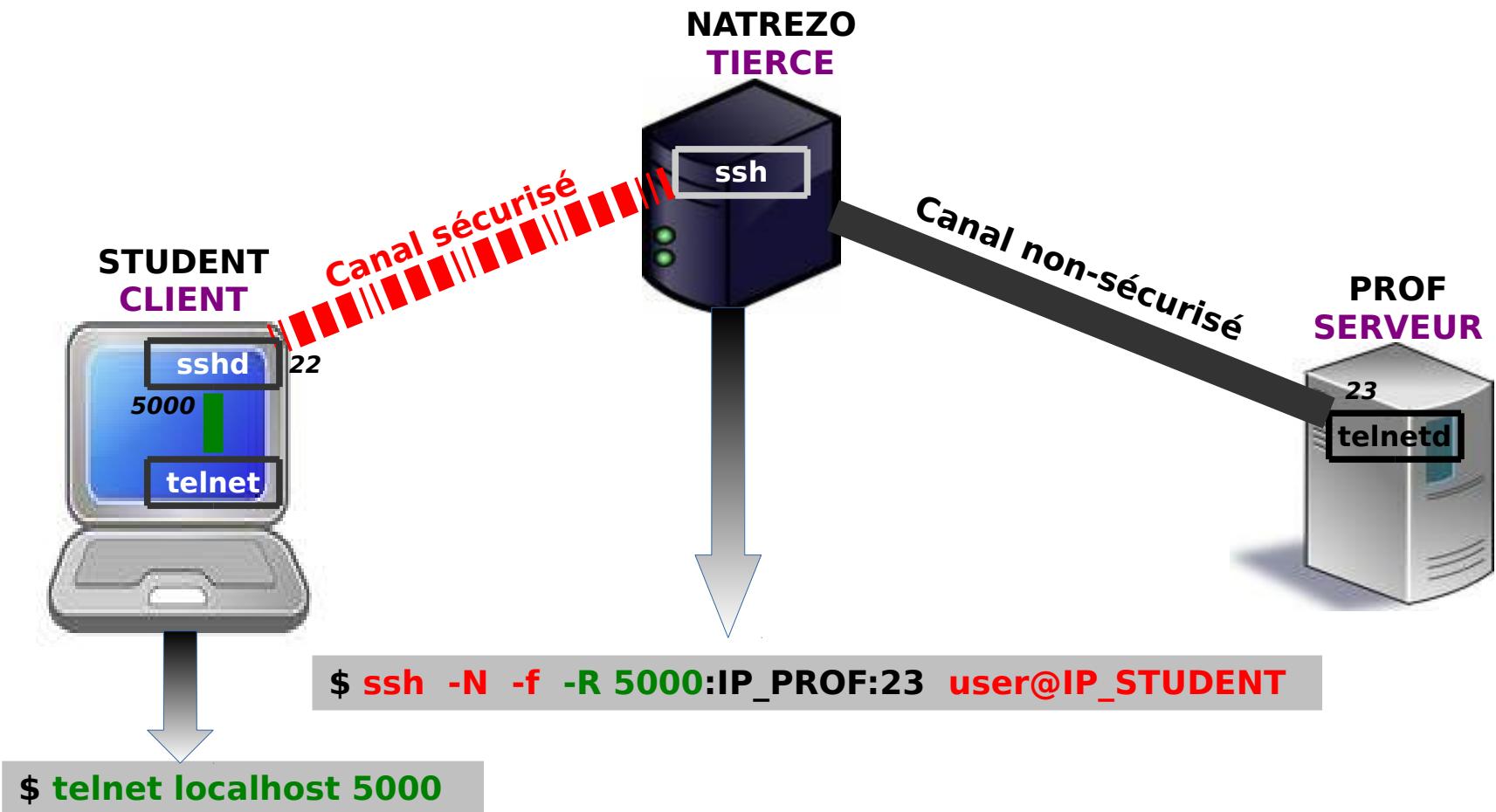
## Remote - Modèle à 2 machines - Constatations

- Un modèle à 2 machines implique l'utilisation de 2 adresses ip différentes dans la commande.
- Le canal reste ouvert même après fermeture de la communication telnet.
- Le client a besoin d'un sshd et d'un compte sur sa machine mais la création du tunnel et son utilisation n'est plus à sa charge.
- Le serveur pourrait créer plusieurs canaux vers des clients différents. Ceux-ci n'auraient plus qu'à les utiliser...



# Le tunneling

Remote - Modèle à 3 machines - Soit 'tunneliser' une conversation telnet...



# Le tunneling

## Remote - Modèle à 3 machines - Constatations

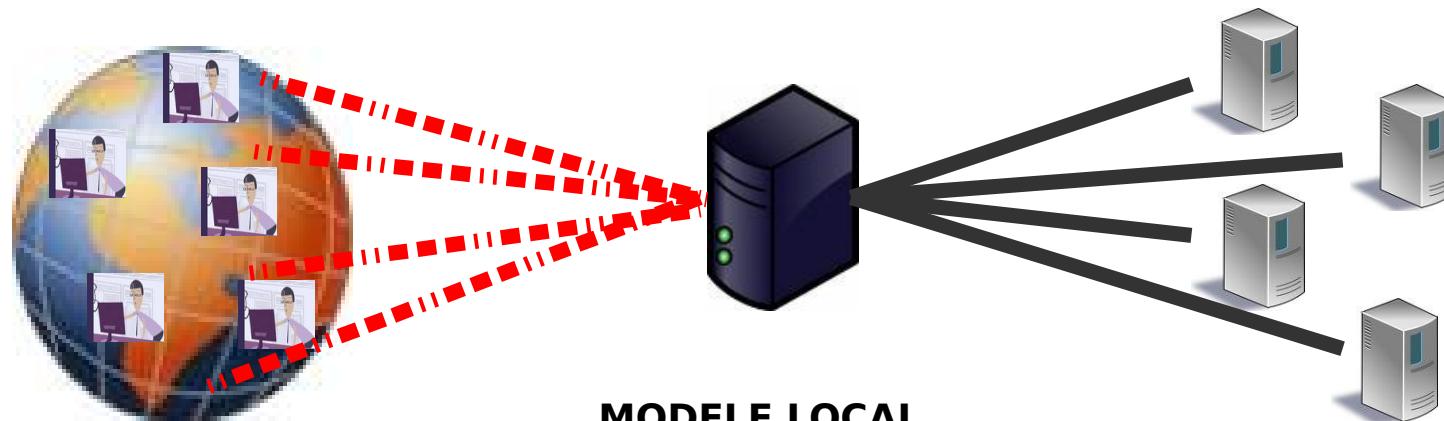
- Un modèle à 3 machines implique l'utilisation des 2 adresses ip différentes dans la commande.
- Tout fonctionne comme si *telnet localhost 5000* pointait directement sur *IP\_PROF:23*
- Le client a besoin d'un sshd et d'un compte sur sa machine mais la création du tunnel et son utilisation n'est plus à sa charge. Néanmoins, il n'a pas besoin d'un compte sur la machine tierce.
- Ici, le sshd et le service à toucher sont sur des machines différentes.
- La communication est chiffrée seulement entre le client et la machine tierce.



# Le tunneling

## Modèle à 3 machines: Remarques

Idéal pour établir, entre plusieurs clients (côté Internet), des communications sécurisées avec plusieurs services d'un Intranet (non chiffré mais considéré comme sûr) via un seul noeud.



Chaque client est à charge de créer son propre tunnel vers la machine tierce sur laquelle il dispose d'un compte Unix.

## MODELE REMOTE

La création de tous les tunnels se fait (bien souvent) par l'administrateur de la machine tierce.



# Le tunneling

- **Autres Remarques**

1° Tunneliser telnet s'avère obsolète car le protocole SSH permet directement d'établir une connexion cryptée...

2° Pourquoi le n° de port relayé est-il supérieur à 1024 ?

- Pour pouvoir les utiliser par la suite sans être root.
- Pour que le port sélectionné ne soit pas un port standard déjà en service.  
Conseil: Vérifier quand même si le port relayé est libre.

3° Le client 'Putty' permet de créer des canaux sécurisés ...



# Le tunneling

- **Atelier**

Réalisez le tunneling d'une session entre un client et un serveur Ftp.

- 1) Avec le client texte ftp sous Linux.
- 2) Avec le client graphique gftp sous Linux.
- 3) Avec le client graphique FileZilla sous Windows.

Bonus:

- a) Reniflez (*tshark*) votre session sans tunneling et avec tunneling.
- b) Constatez quels sont les ports ouverts sans et avec tunneling.



# Références

- **WEBOGRAPHIE**

<http://www.openssh.com/fr/index.html>

<http://fr.wikipedia.org/wiki/Ssh>

<http://www.commentcamarche.net/contents/crypto/crypto.php3>

<http://www.linux-france.org/prj/edu/archinet/systeme/index.html>

<http://www.authsecu.com/cours-formation-elearning/cours-formation-elearning.php>

<http://www.ac-creteil.fr/reseaux/systemes/linux> (TP ssh)

- **BIBLIOGRAPHIE**

SSH, le shell sécurisé - La référence  
Par Daniel-J Barrett,Richard-E Silverman  
(Edition Oreilly) - *Dispo sur Google books*



## Aspects Réseaux sous Linux

# LE SYSTEME DNS



**Jean-Louis Gouwy**



# Plan

- But et historique
- Plan du DNS
  - Système de nommage / Domaine / Zone / Obtenir un nom de domaine / Autorité / Délégation / Serveurs racines / Redondance
- Résolution et résolution inverse
- Parcours d'une requête
- Debug DNS
- Serveur de cache
  - Architecture / Utilités / Configuration minimale / Serveur forward
- Bind
  - Présentation / Packages / Composants / Serveur cache / Serveur forward esclave
- Exercices: Serveur de cache
  - Exercice 1: Serveur de cache / Exercice 2: Serveur forward esclave
- Serveur autoritaire
  - Serveur primaire / Serveur secondaire / Architecture / Remarques
- Base de données
- Bind
  - Gérer un domaine / Serveur autoritaire récursif / Serveur autoritaire itératif / Les fichiers de zone



# Plan (suite)

- Exercice: Serveur autoritaire
  - Exercice 3: Serveur autoritaire de cache
- Délégation et sous domaine
- Exercice: Délégation et sous domaine
  - Exercice 4
- Redondance
- Bind
  - Configuration du serveur primaire / Configuration du serveur secondaire /
  - Synchronisation
- Exercice: Redondance
  - Exercice 5
- Références

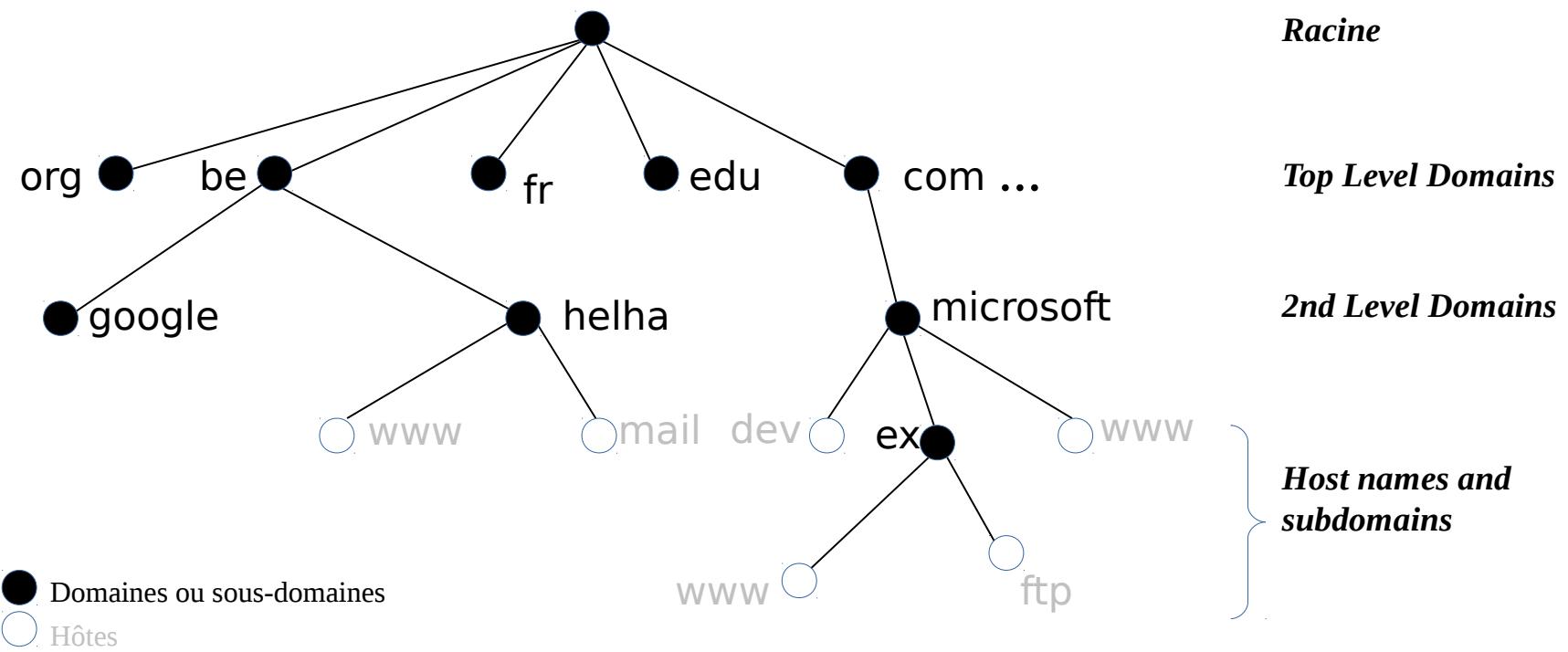


# But et historique

- Pour traduire des noms DNS (FQDN - Fully Qualified Domain Name) en adresse IP et l'inverse
  - [www.helha.be](http://www.helha.be) → 193.190.66.18
  - 193.190.66.18 → [www.helha.be](http://www.helha.be)
- Service essentiel pour tout réseau relié au monde extérieur
- Très utile aussi pour un réseau local (remplacement de /etc/hosts)
- Car le fichier /etc/hosts est très limité
  - doit être recopié sur toutes les machines
  - mises à jour, ajouts et suppressions synchronisées fastidieuses
- En 1984, Paul Mockapetris mit au point un système de nommage: le DNS (décrit dans les RFC 883 et 884, puis 1034 et 1035).



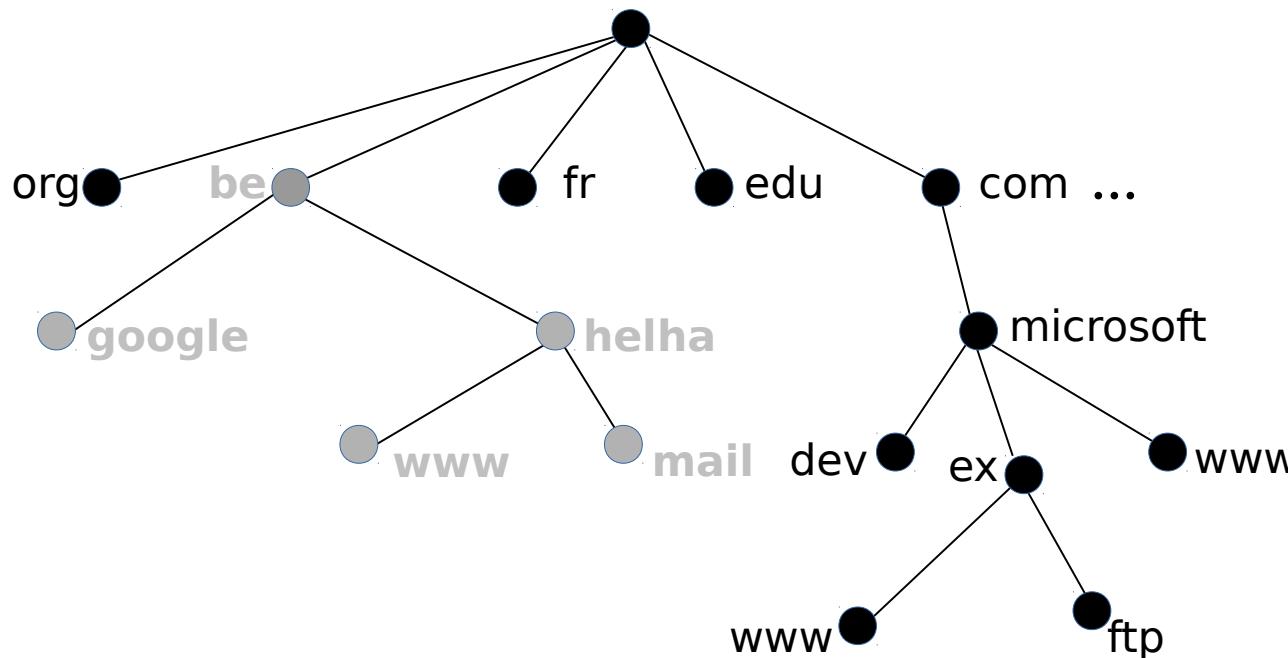
# Plan du DNS: Système de nommage



- Basé sur un modèle en arborescence similaire à celui des systèmes de fichiers.
- La dénomination d'un nom commence par le bas pour se terminer à la racine.  
(ex. [ftp.ex.microsoft.com](ftp://ftp.ex.microsoft.com). → le dernier point étant optionnel)



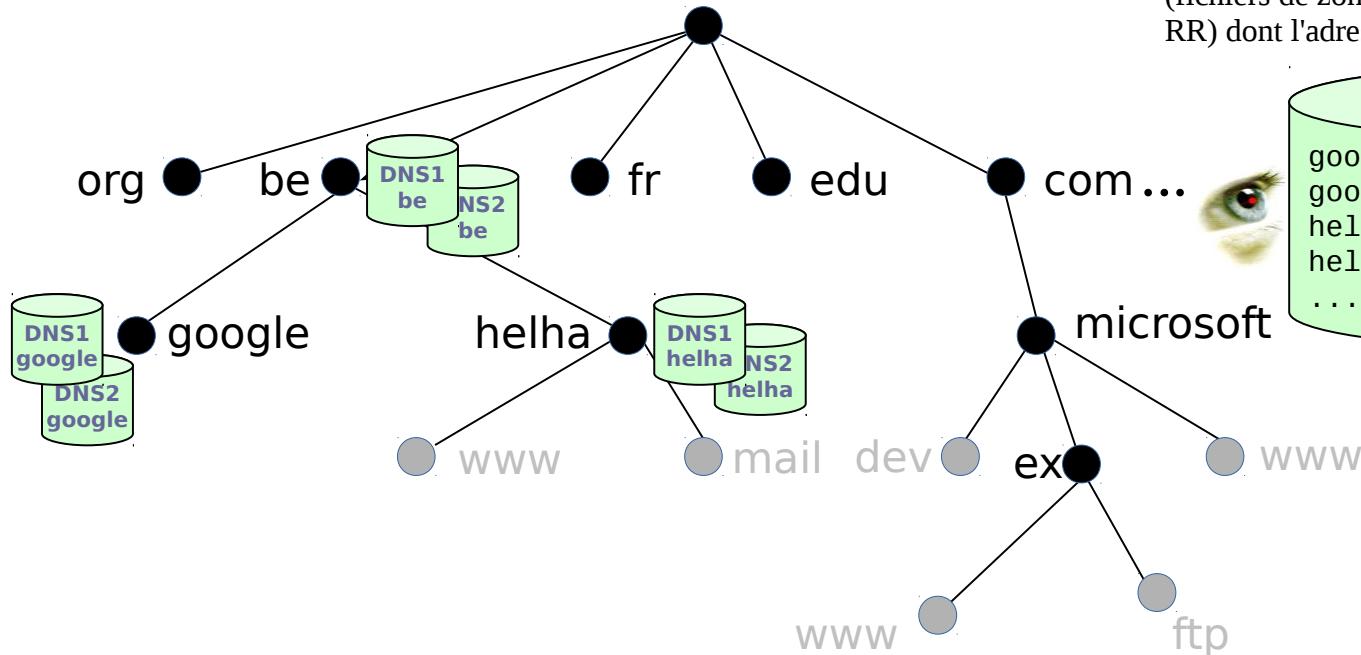
# Plan du DNS: Domaine



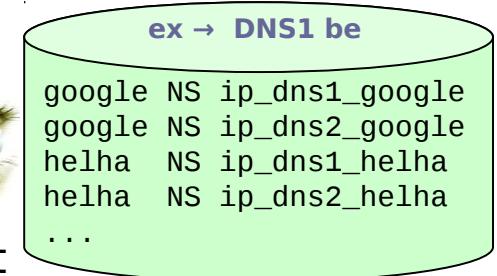
- Un domaine est l'ensemble d'une sous arborescence.  
Exemple: Le domaine "be." rassemble toute la sous-arborescence à partir du noeud be.
- Il y a un domaine racine " ." et des domaines fils: ex. "org."
- Un nom de domaine est utilisé dans les URL et les adresses de messagerie (ex. Soit le domaine "helha.be" → URL : <http://www.helha.be>  
→ @ : toto@helha.be)



# Plan du DNS: Zone



Chaque nœud contient une (ou plusieurs) BD (fichiers de zone) stockant des informations (les RR) dont l'adresse des BD des fils.



⚠ Pour que les RR de type NS soient propagés correctement dans les TLD, il est nécessaire d'enregistrer le nom de domaine (voir ci-après).

- Une zone est la partie descriptive pour un niveau donné.
- Elle est restreinte à un nœud → une zone est constituée de la base de données décrivant un nœud.
- ⚠ Un RR (Ressource Record) de type NS renseigne l'adresse Ip d'un serveur Dns d'un sous-domaine.



# Plan du DNS: Obtenir un nom de domaine

C'est ICANN (<https://www.icann.org/fr>) qui gère les TLD et délègue leur gestion technique à différents opérateurs (*registry*).

ex. DNS BELGIUM (<https://www.dnsbelgium.be/fr>) pour le domaine .be notamment.

Chaque registry peut délèguer à plusieurs bureaux d'enregistrement (*registrar*) la vente des noms de domaine.

ex. DNS BELGIUM travaille notamment avec les bureaux d'enregistrement BELNET (Belgique), OVH (France), GANDI (France)...

(<https://www.dnsbelgium.be/fr/nom-de-domaine/trouvez-un-agent#/list/tld/be>)



# Plan du DNS: Obtenir un nom de domaine

## Procédure d'obtention d'un nom de domaine:

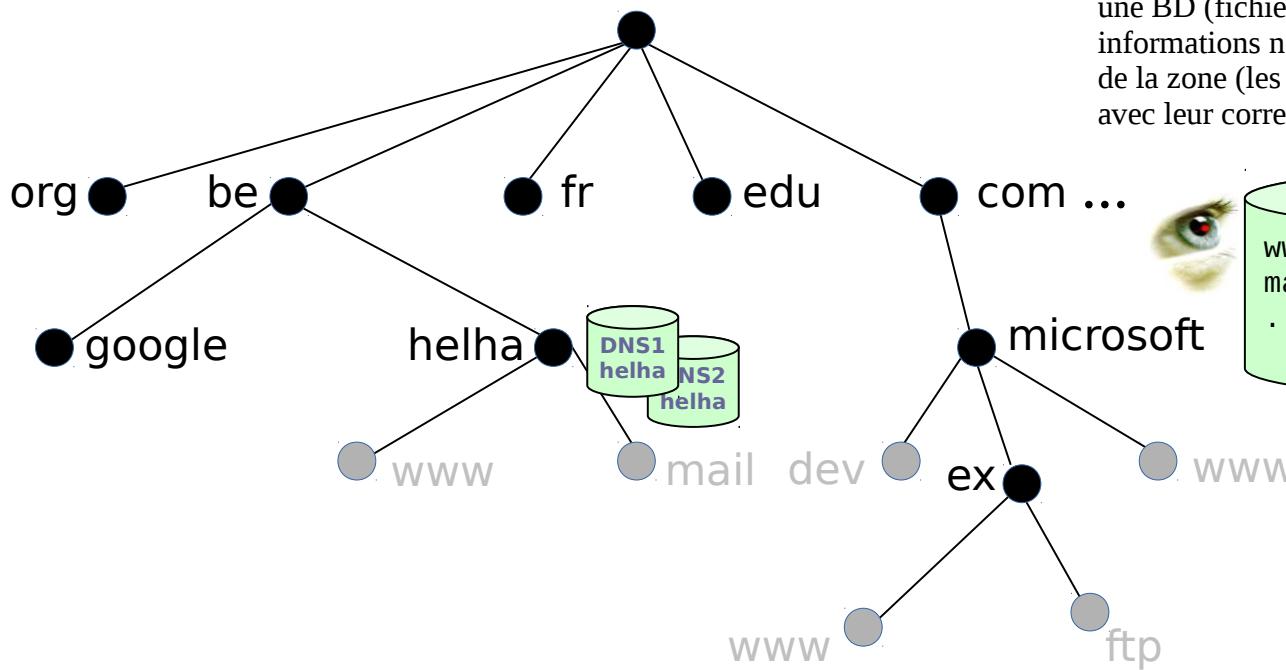
- Aller sur le site web d'un registry ou registrar
- Utiliser l'outil 'whois' en ligne (disponible également en ligne de commande sous Linux) afin de vérifier si le nom de domaine est disponible.
- L'enregistrer (le louer): remplir un formulaire avec différentes informations et détails.

## 2 cas de figure:

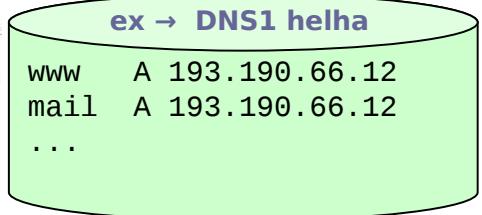
- . Vous gérez vos propres serveurs DNS:  
dans ce cas vous donnez au registrar l'Ip de vos serveurs DNS qui gèrent le domaine que vous avez acheté.
  - . Le bureau d'enregistrement gère votre DNS sur ses serveurs:  
dans ce cas vous utilisez en général une interface web simplifiée fournie par le bureau d'enregistrement.
- Le bureau d'enregistrement vous informera dès que le nom de domaine sera enregistré et vous pourrez alors l'utiliser



# Plan du DNS: Autorité



Chaque nœud a autorité sur la zone et contient une BD (fichiers de zone) stockant les informations nécessaires au bon fonctionnement de la zone (les RR) dont les noms de machines avec leur correspondance IP.

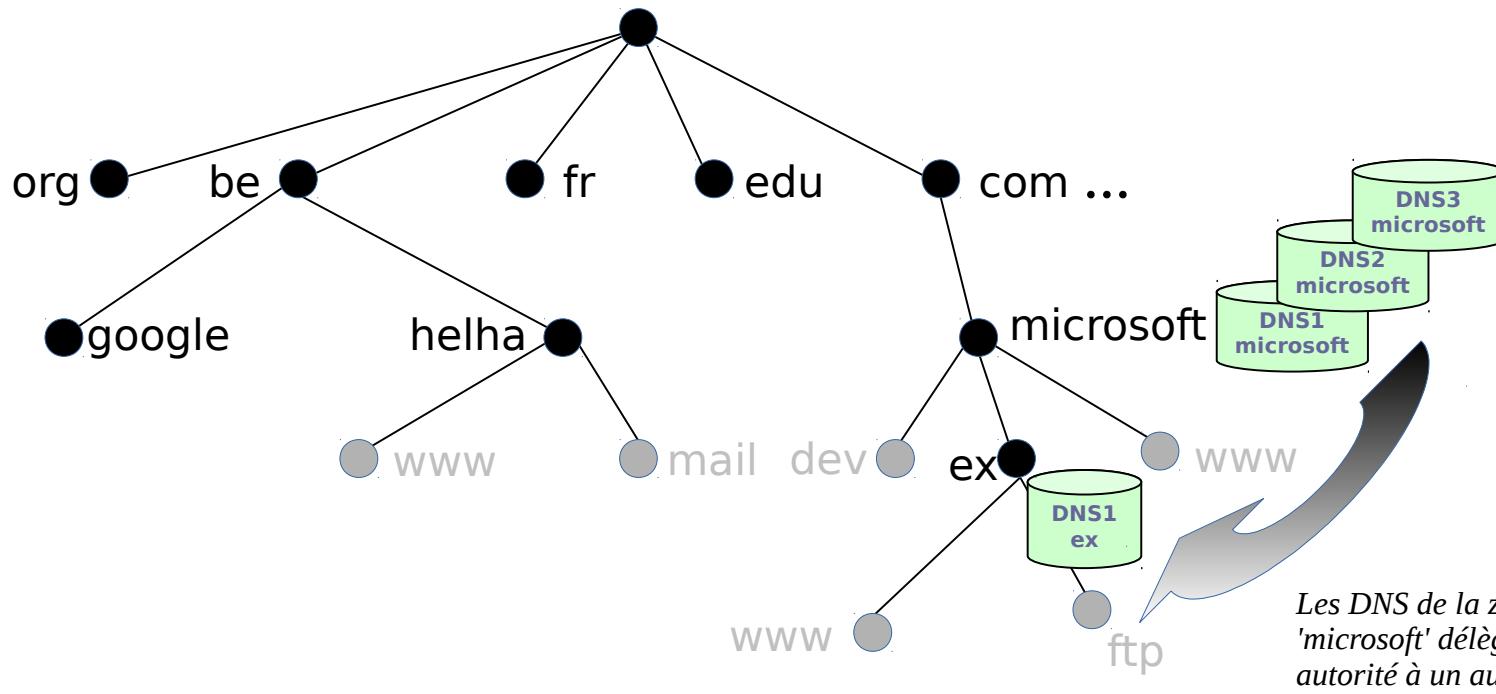


↗ Ici, on constate que les hosts `www.helha.be` et `mail.helha.be` possèdent la même adresse IP.

- ☞ Un DNS particulier s'occupe d'un nœud sur lequel il a autorité. On dit que le serveur gère une zone d'autorité.  
C'est à dire qu'il gérera l'attribution des noms et résoudra les noms via un fichier de zone (BD) distinct pour chaque nœud.
- ☞ Un RR de type A établit une correspondance entre un nom de host et son adresse Ip.



# Plan du DNS: Délégation

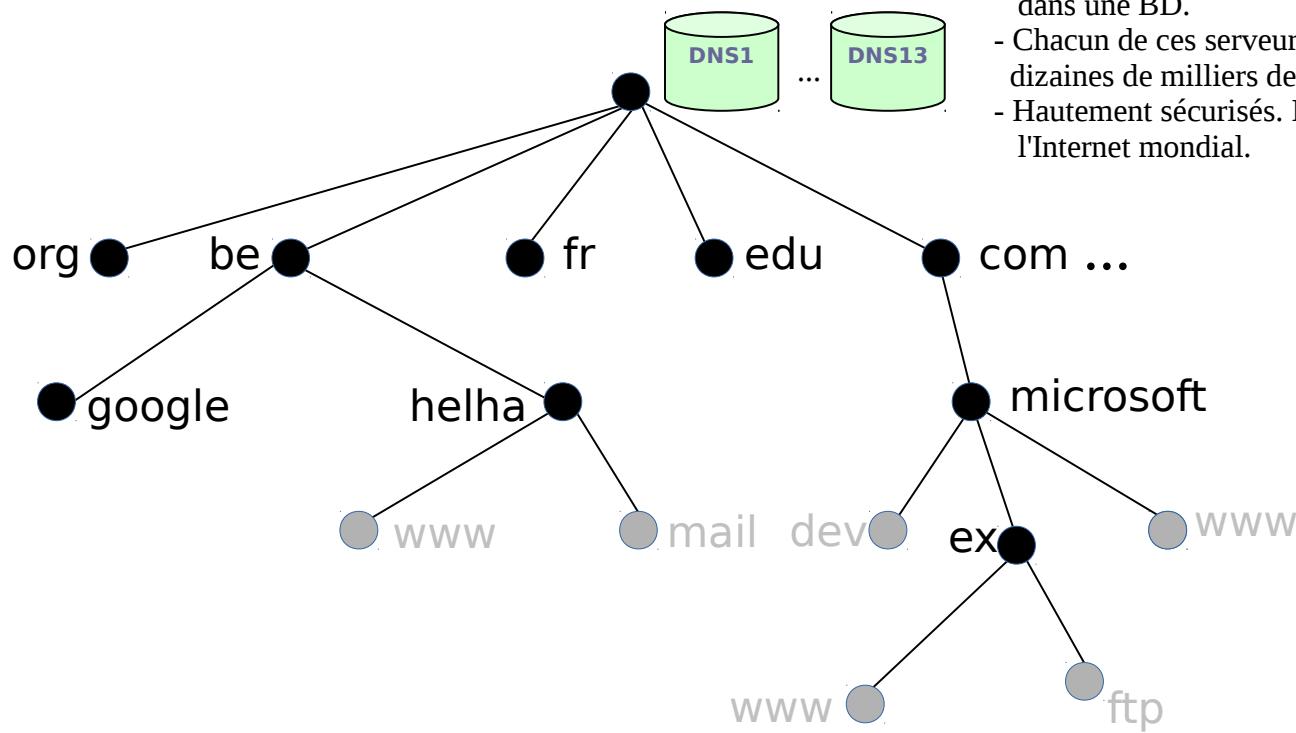


*Les DNS de la zone 'microsoft' délèguent leur autorité à un autre DNS en ce qui concerne la gestion de la zone 'ex'.*

- Un serveur faisant autorité sur une zone peut déléguer la gestion de ses sous-domaines à d'autres serveurs de nom.
- Des fichiers de zone pour chaque sous-domaine doivent donc être créés et les fichiers de zone du domaine parent devront être modifiés en conséquence.



# Plan du DNS: Serveurs racines



- 13 serveurs répartis dans le monde et gérés par 12 organisations indépendantes, chacun contenant les références de tous les serveurs de premier niveau dans une BD.
- Chacun de ces serveurs est consulté plusieurs dizaines de milliers de fois par heure.
- Hautement sécurisés. Il en va de l'utilisation de l'Internet mondial.

💡 Tout serveur DNS doit connaître les adresses IP des serveurs racines.



# Plan du DNS: Serveurs racines (suite)

- Ils contiennent la même information grâce à un système de réPLICATION.
- Ils sont identifiés par les lettres de A à M et appartiennent tous au même domaine ROOT-SERVERS.NET.



# Plan du DNS: Serveurs racines (suite)

- Plusieurs de ces serveurs correspondent à plusieurs serveurs répartis dans le monde:<http://root-servers.org/>

The screenshot shows the 'Root Servers' page with the 'K' server selected. The interface includes a navigation bar with letters A through M, a sidebar with sections for Operator (RIPE NCC), Locations (Sites: 56), and IP (IPv4: 193.0.14.129, IPv6: 2001:7fd::1). The main content area lists various global locations where the K root server is mirrored.

Location
Amsterdam, NL
Ashland, US
Athens, GR
Barcelona, ES
Beirut, LB
Belgrade, RS
Berlin, DE
Brisbane, AU
Bucharest, RO
Budapest, HU
Buenos Aires, AR
Calgary, CA
Cork, IE
Doha, QA
Frankfurt, DE
Gdynia, PL
Geneva, CH
Hamburg, DE
Helsinki, FI
Johannesburg, ZA
Kansas City, US
Karlsruhe, DE
Kuwait City, KW
London, UK
Miami, US
Milan, IT
Montevideo, UY
Moscow, RU
Mumbai, IN
Noida, IN
Novosibirsk, RU
Ottawa, CA
Paris, FR
Poznan, PL
Prague, CZ
Reno, US
Reykjavik, IS
Richmond, US
Riga, LV
Saint Petersburg, RU
San Jose, CR
Santiago, CL
Semey, KZ
Sofia, BG
St George, US
Tbilisi, GE
Tehran, IR
Tokyo, JP
Vienna, AT
Yerevan, AM
Zurich, CH

- Ici, le serveur K.ROOT-SERVERS.NET, accessible par une adresse Ipv4 et Ipv6 et dupliqué 56 fois dans le monde, est géré par l'organisation RIPE NCC.
- Le serveur A.ROOT-SERVERS.NET est le serveur d'origine. Les autres (B → M) sont des serveurs miroirs de celui-ci.



# Plan du DNS: Serveurs racines (suite)

- Un serveur racine est en fait constitué d'un ensemble de serveurs dupliqués et configurés en 'anycast'. Autrement dit, chacun de ceux-ci possèdent la même adresse Ip mais un seul répondra à la requête DNS.

Lorsqu'un routeur reçoit une demande pour joindre une adresse 'anycast', il la route généralement vers le serveur géographiquement le plus proche.

→ nécessité de pouvoir configurer des serveurs DNS anycast et des routeurs supportant un protocole de routage dynamique - ex. BGP (Hors cadre du cours)

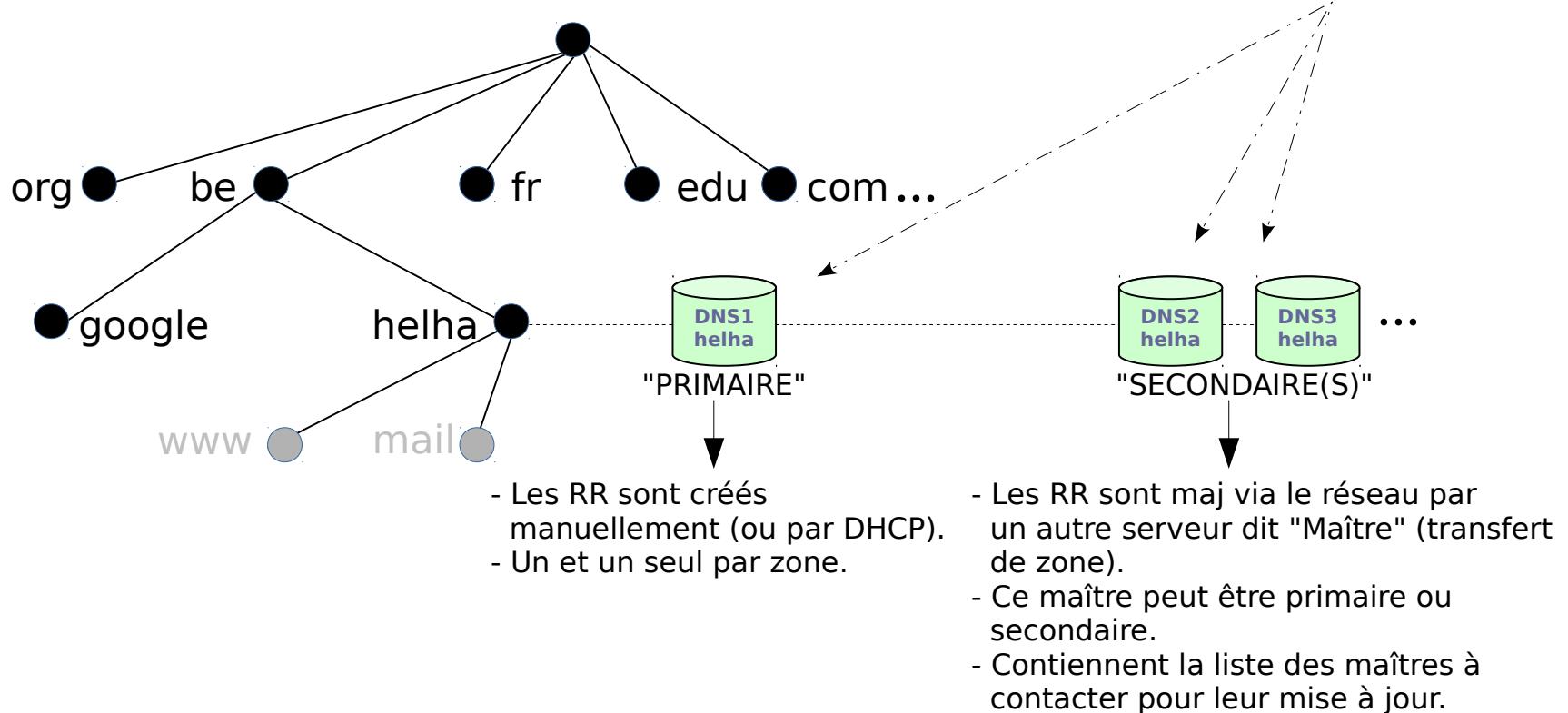
Plus d'info. sur l'anycast et le protocole BGP :

<https://vincent.bernat.im/fr/blog/2011-dns-anycast>



# Plan du DNS: Redondance

Dans la réalité, plusieurs serveurs de noms font très souvent autorité pour un même domaine.

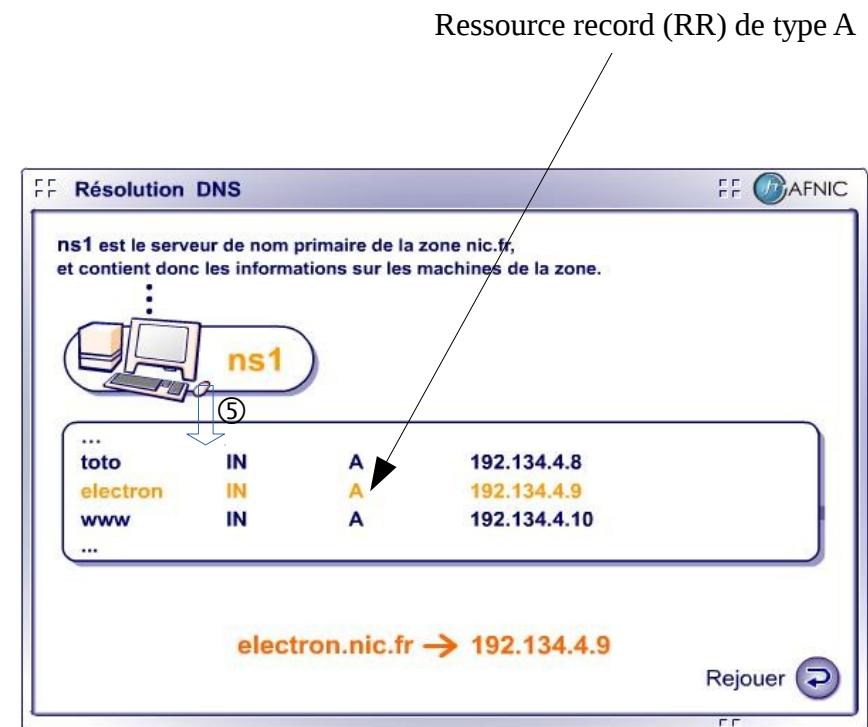
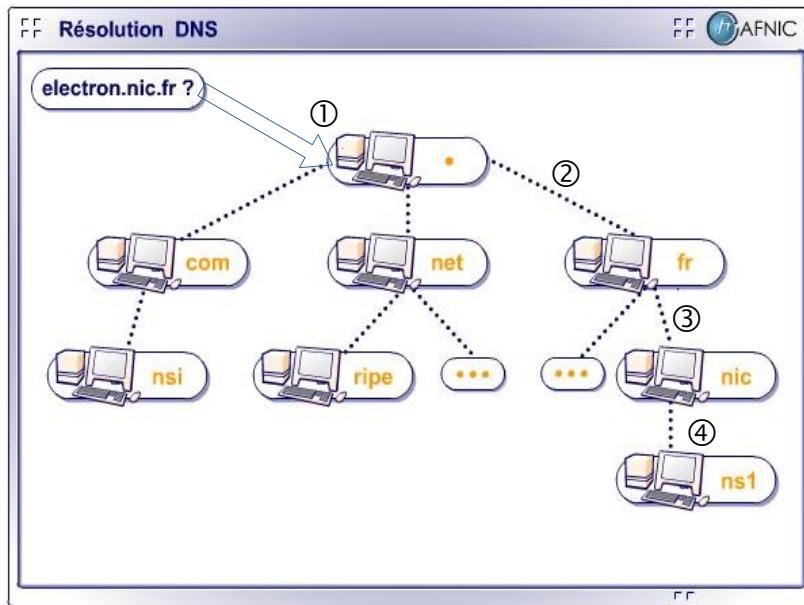


- Backup (meilleure tolérance aux pannes)
- Loadbalancing (répartition des charges possible)



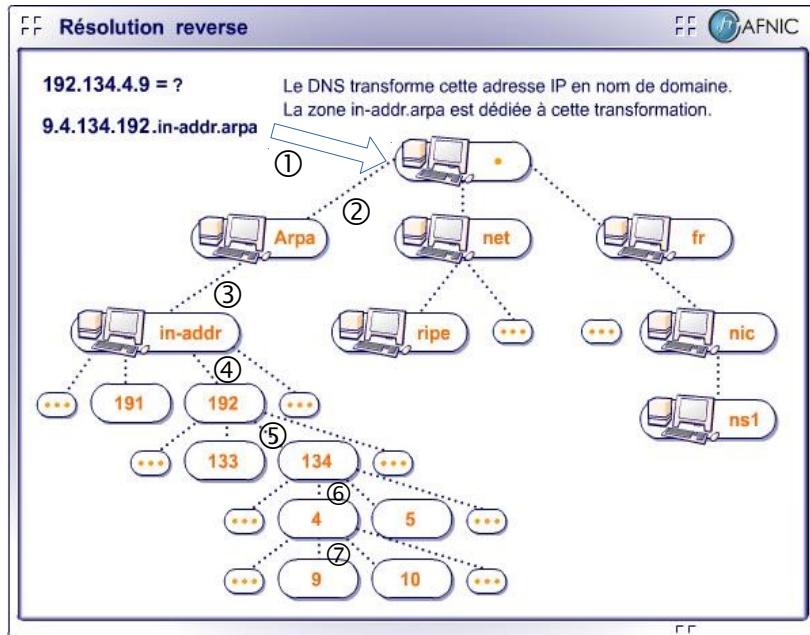
# Résolution et résolution inverse

- Comment fait le DNS pour retrouver une adresse IP à partir d'un nom de machine ?

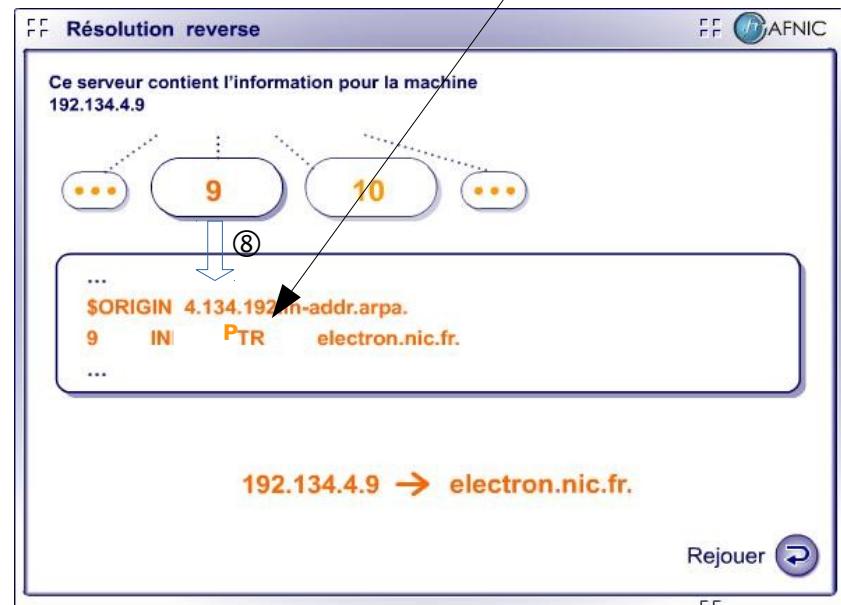


# Résolution et résolution inverse

- Comment fait le DNS pour retrouver un nom de machine à partir d'une adresse Ip ?



Ressource record (RR) de type PTR



# Résolution et résolution inverse

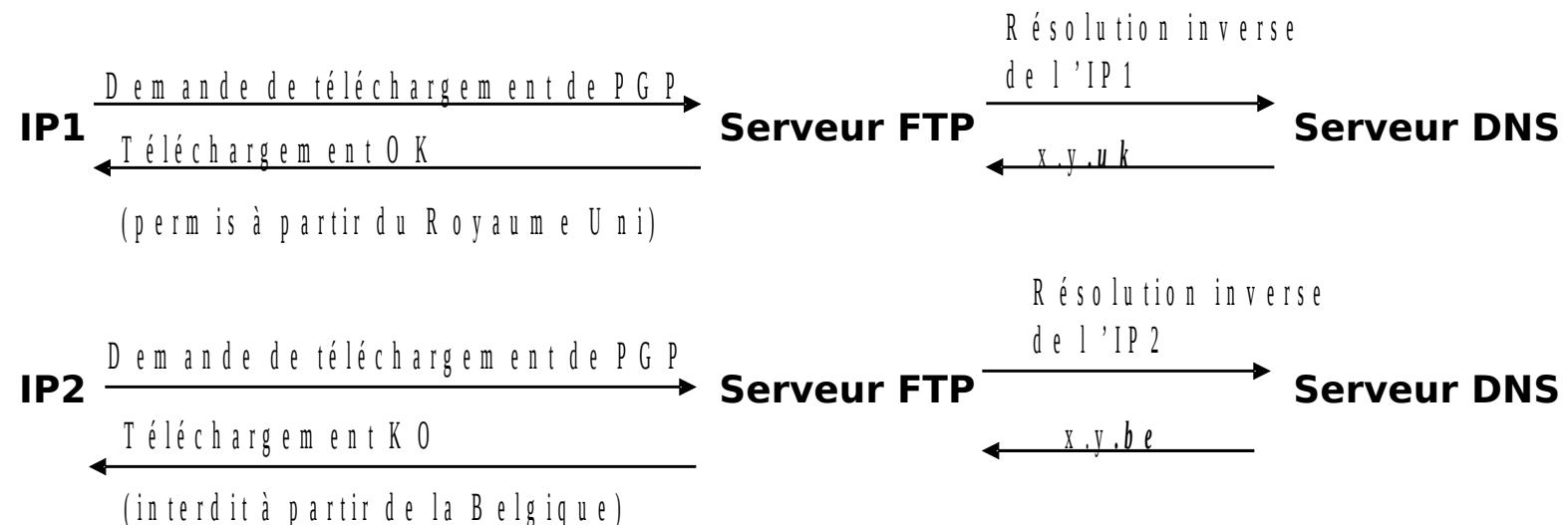
## Résolution inverse: Utilités

- Utile pour restreindre l'accès à des services Internet, permettre les règles anti-spam des serveurs de messagerie ...
- Exemple

Soit un logiciel de cryptographie PGP installé sur un serveur FTP.

FTP est configuré pour que ce logiciel ne soit téléchargeable qu'à partir de certains pays ...

La résolution inverse sera donc sollicitée par le serveur FTP au serveur DNS.



# Parcours d'une requête

- › Le "resolver" permet de communiquer avec les serveurs DNS
- › 2 modes d'interrogation:

Récursif: Le client envoie une requête à un serveur, ce dernier devant interroger tous les autres serveurs nécessaires pour renvoyer la réponse complète au client (mode utilisé par les machines clientes en général).

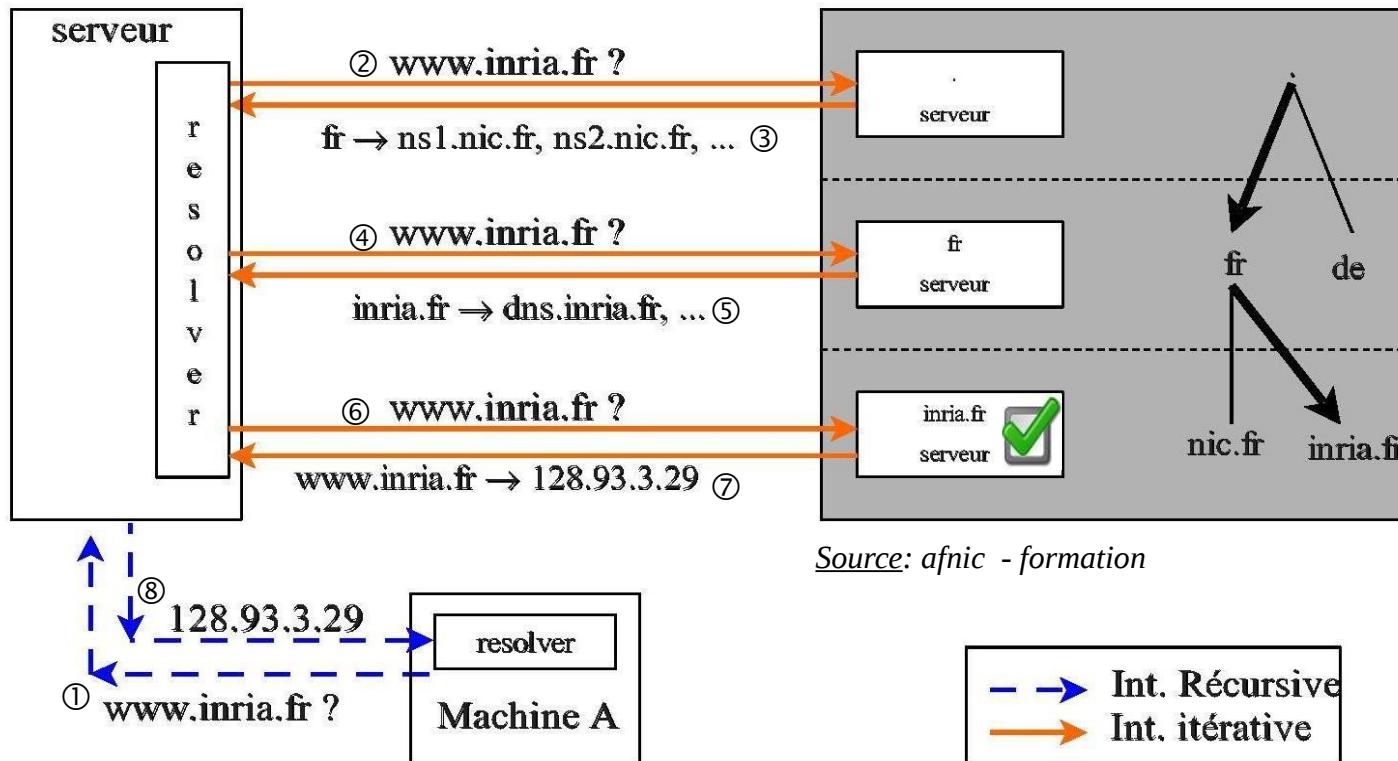
Une requête récursive attend une réponse définitive à une résolution de noms.

Itératif: Le client envoie une requête à un serveur, ce dernier renvoyant la réponse si il la connaît, ou le nom d'un autre serveur qu'il suppose plus renseigné pour résoudre cette question (mode utilisé par le resolver des serveurs en général).

Une requête itérative attend pour réponse la réponse elle-même ou bien une référence vers un autre serveur DNS.



# Parcours d'une requête



- Ici, il y a eu 4 interrogations pour résoudre `www.inria.fr`
- Mécanisme accélérateur: le cache (voir plus loin)



# Parcours d'une requête

## **Remarque: DNS récursif ouvert**

- Il est fortement conseillé de ne pas laisser votre DNS récursif ouvert.
  - C'est-à-dire ne pas permettre la récursion sur votre DNS à partir d'Internet.
  - Votre DNS n'acceptera de résoudre des noms qu'à partir votre réseau local.
- 
- 😊 Ne pas être DNS relais.  
Accroissement de la sécurité (hors cadre du cours)



# Debug DNS

**dig (Domain Information Groper)**

*man dig*

# dig → donne la liste des serveurs racines

# dig @server name type → donne les informations concernant une ressource (name) d'un certain type (type) d'un certain serveur Dns (@server) .

## Atelier 1 (dig):

Suivre la chaîne des délégations entre les zones à partir de la racine jusqu'à l'atteinte du nom Dns demandé soit ici www.reseaucerta.org pour connaître son ip.

# dig @e.root-servers.net www.reseaucerta.org A → on choisit un serveur racine

...

# dig @d0.org... www.reseaucerta.org A → on choisit un serveur ayant autorité sur org.

# dig @a.dns.gandi.net www.reseaucerta.org A → on choisit un serveur ayant autorité sur reseaucerta.org.

www.reseaucerta.org 86400 IN A 194.254.4.9 → adresse ip recherchée



# Debug DNS

## Atelier 2 (dig):

Toujours en suivant la chaîne des délégations entre les zones à partir de la racine, tentez de résoudre un nom qui n'existe pas.

```
# dig @e.root-servers.net www.axyzz.be A → on choisit un serveur racine  
...  
  
# dig @brussels.ns.dns.be www.axyzz.be A → on choisit un serveur ayant autorité  
... sur be.  
...  
;; ->> HEADER ... status : NXDOMAIN → Non existant DOMAIN
```

## Autres cas

```
# dig www.helha.be  
# dig +trace www.helha.be → Recherche à partir de la racine  
# dig lesoir.be MX  
# dig -x 204.13.162.123 → Recherche inverse
```



# Debug DNS

## nslookup

*man nslookup*

Cette commande peut être utilisée en mode interactif

### Atelier 3 (nslookup):

Suivre la chaîne des délégations entre les zones à partir de la racine jusqu'à l'atteinte du nom Dns demandé soit ici [www.reseaucerta.org](http://www.reseaucerta.org) pour connaître son ip.

```
# nslookup
> server e.root-servers.net      → on choisit un serveur racine
> set type=NS                   → on s'intéresse aux records de type NS
> org.                          → quels sont les dns qui gèrent org. ?
...
> server d0.org...              → on passe sur un de ces serveurs
> reseaucerta.org.              → quels sont les dns qui gèrent reseaucerta.org. ?
...
> server a.dns.gandi.net       → on passe sur un de ces serveurs
> set type=A                    → on s'intéresse aux records de type A
> www.reseaucerta.org          → quelle est l'Ip de www.reseaucerta.org ?
```



# Debug DNS

```
> set type=MX                                → on s'intéresse aux records de type MX  
> reseaucerta.org.                           → quels sont les serveurs de mail de reseaucerta.org ?  
...  
> set type=A                                → on s'intéresse aux records de type A  
> smtp.reseaucerta.org.                      → quelle est l'ip du serveur smtp de reseaucerta.org ?  
...  
> set type=ANY                               → on s'intéresse à tout  
> reseaucerta.org.  
...
```

## Autres cas

```
# nslookup www.lelibre.be → C'est le Dns par défaut qui est choisi pour résoudre le nom.  
# nslookup www.lelibre.be 8.8.8.8 → C'est un autre Dns qui est choisi.  
# nslookup 37.59.106.19                  → Recherche inverse résolue avec le Dns par défaut.
```



# Debug DNS

## Atelier 4 (nslookup):

Toujours en suivant la chaîne des délégations entre les zones à partir de la racine, tentez de réaliser une recherche inverse (soit résoudre 91.121.208.164)

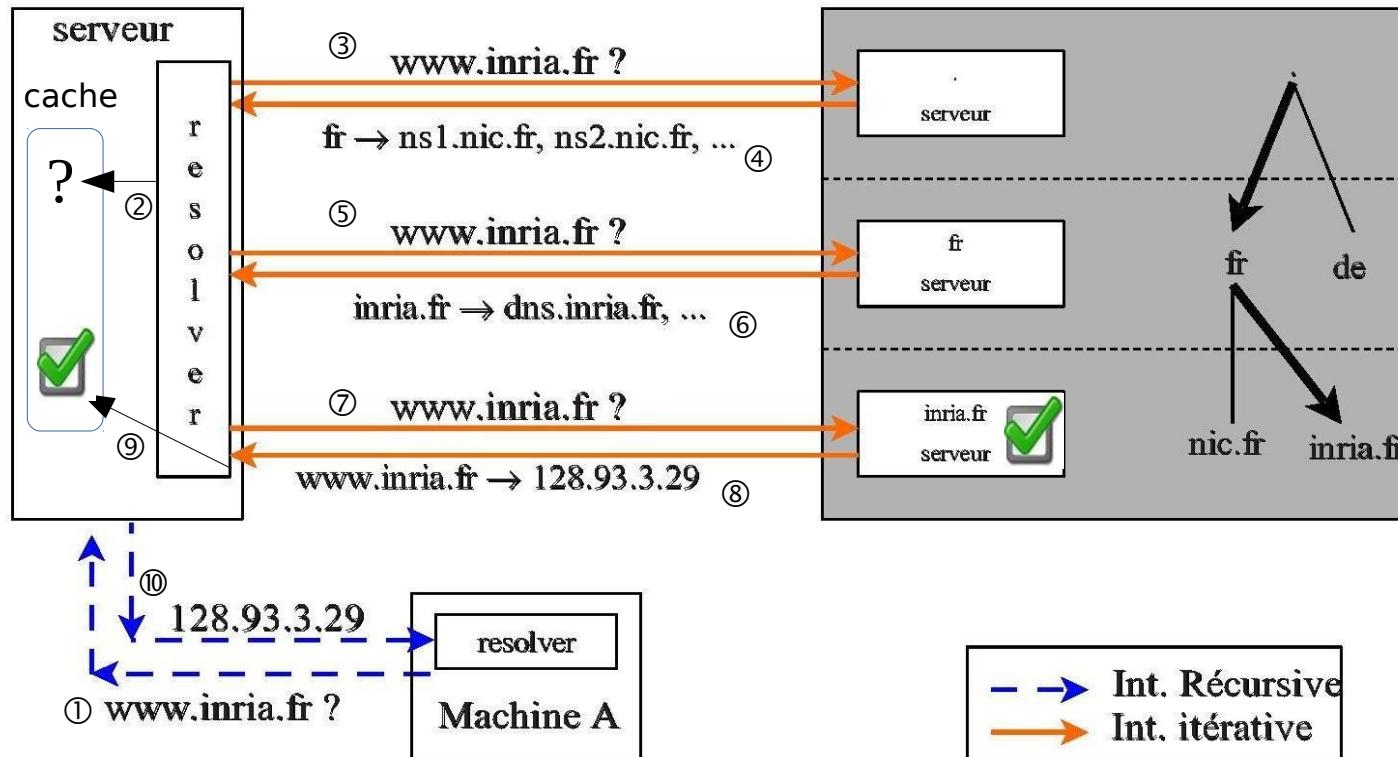
```
# nslookup
> server e.root-servers.net
> set type=PTR
> 91.121.208.164
...
...
...
> server dns12.ovh.net
> 91.121.208.164
...
...
```

- *on choisit un serveur racine*
- *on s'intéresse aux records de type PTR*
- *y a-t-il un RR de ce type dans sa zone in-addr.arpa ?*
- *Et le serveur ne me répond pas directement, mais m'envoie une liste de serveurs ont autorité sur les adresses qui commencent par 91*
- *on en choisit un au hasard ...*
- *... et on lui repose la même question*
  
- *Ici, on a déjà la réponse. Cela signifie que dns12.ovh.net a autorité sur toutes les Ips du domaine 91.in-addr.arpa car il s'agit d'un réseau de classe A*



# Serveur de cache

## Architecture

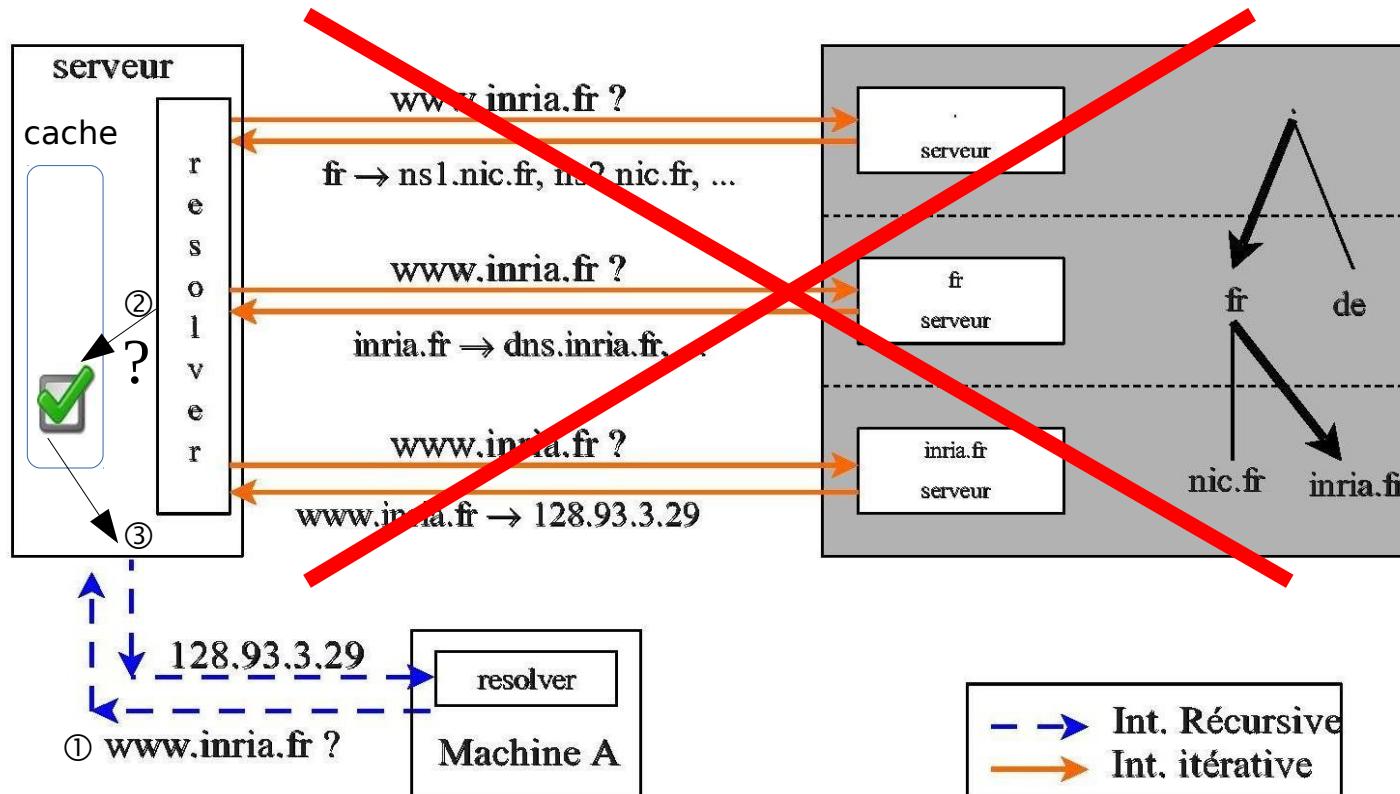


- Ici, le resolver du dns interroge d'abord son cache
- Si celui-ci est vide, il interroge alors les dns extérieurs de manière itérative ou fera appel à des 'forwarders' afin de résoudre la requête (voir plus loin)
- Une fois résolue, l'association Ip/Nom est mise en cache



# Serveur de cache

## Architecture: suite



- Lorsque la réponse à la requête est déjà cache, l'extérieur n'est pas sollicité.
- Le resolver du dns la transmet au client à partir de ce cache.  
Il s'agit d'une Non-authoritative answer.



# Serveur de cache

## Avantages

- éviter la surcharge inutile du réseau
  - supprimer les délais du réseau
  - amoindrir la charge des autres serveurs
- tout serveur possède en général au minimum un cache

## Inconvénient

- ne pas oublier de sécuriser le cache



# Serveur de cache

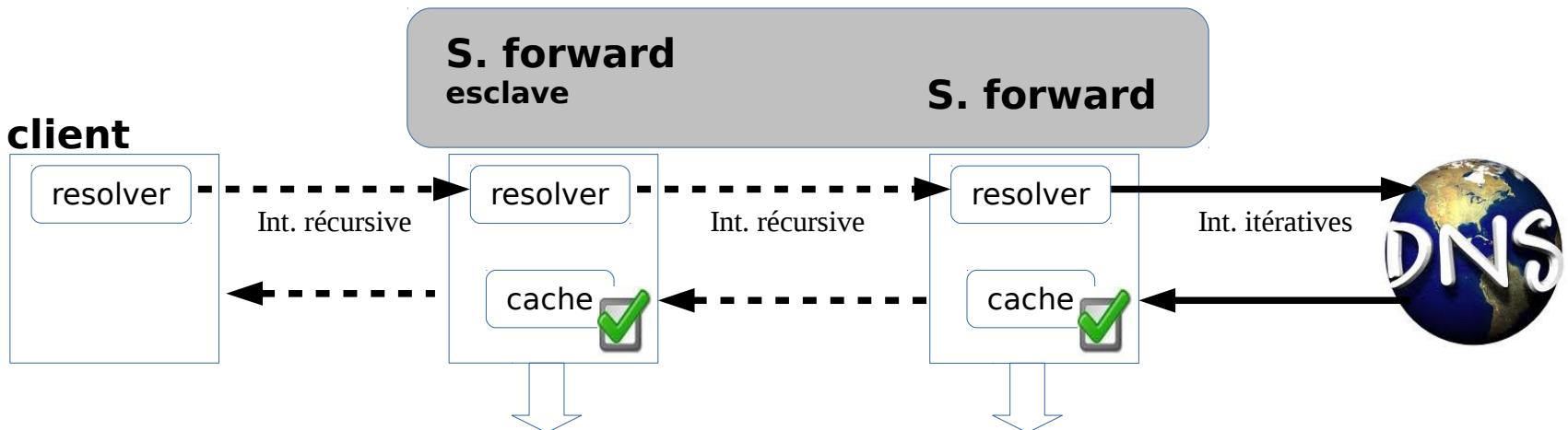
## Configuration minimale

- Base de données nécessaire
  - adresses des serveurs de la racine
  - reverse du loopback 1.0.0.127.in-addr.arpa (c'est elle qui fait office de zone de cache)
- Les données du cache possèdent une durée de vie limitée (Time To Live – ttl) afin de permettre son rafraîchissement et la prise en compte des modifications.
- Il s'enrichit au fur et à mesure par les données récoltées pour résoudre les requêtes des clients.
  - une requête déjà demandée est résolue à partir du cache du serveur
- Ce type de serveur n'a autorité sur aucune zone.



# Serveur de cache

## Serveur forward



- Si la requête soumise ne peut être résolue par le cache, il la fait suivre à un autre serveur dit "faire-suivre" ou "forward".
- Si échec avec tous les serveurs forward, il renvoie une erreur.
- Il possède donc une liste de serveurs forward.
- Il s'agit souvent d'un serveur interne.

- Si la requête soumise ne peut être résolue par le cache, il tente de résoudre lui-même la requête (ou fait suivre s'il possède une liste de serveurs forward).
- Il ne possède pas toujours une liste de serveurs forward.
- Il s'agit souvent des serveurs du FAI.

## Utilité

En interne, les clients utilisent x serveurs forward <sub>esclaves</sub> lesquels font suivre vers un seul serveur forward.

=> enrichissement rapide d'un cache partagé et commun sur le serveur forward.



# Bind

## Présentation

- Bind (Berkeley Internet Name Daemon)
- Serveur de noms le plus utilisé sur Internet.
- Bind 9 supporte l'IPv6, les noms de domaine unicode, le multithread et de nombreuses améliorations de sécurité.
- Maintenu actuellement par Paul Vixie avec l'Internet Software Consortium.  
<http://www.isc.org>
- Dernière version stable: Bind 9.11



# Bind

## Packages

- bind – bind-utils – bind-libs
- bind-chroot: pour travailler dans un environnement "chrooté" (hors cadre)

## Composants

- Le daemon *named*
  - c'est le service Dns → # service named start/stop/restart
  - port d'écoute (udp 53)
- Fichiers de configuration
  - */etc/named.conf* → fichier de configuration principal
  - */var/named* → dossier par défaut qui contient les fichiers de zones
- Outils de debuggage:  
*dig, nslookup, host ...* (inclus dans le package *bind-utils*)
- Points d'entrée: *man named, man /etc/named.conf*



# Bind

## Serveur cache

### named.conf

```
options {
    listen-on port 53 {127.0.0.1 ; 192.1.0.2 ; } ; // Port d'écoute, ip admises
    directory "/var/named"; // répertoire des fichiers de zones
};

zone "." IN {
    type hint;
    file "named.ca"; // cache des serveurs racines
};

zone "1.0.0.127.in-addr.arpa" IN {
    type master;
    file "named.loopback"; // zone primaire du reverse loopback
};

zone "localhost" IN {
    type master;
    file "named.localhost"; } // Zone primaire du loopback.
                                // Facultative sauf si on veut faire
                                // résoudre le nom 'localhost' par le serveur.
```

Ces 3 fichiers de zones seront automatiquement créés lors de l'installation du package 'bind'...



# Bind

## Serveur forward esclave

### named.conf

```

options {
    listen-on port 53 {127.0.0.1 ; 192.1.0.2 ;} ;
    directory "/var/named";
    forwarders {192.2.0.3 ; 192.3.0.3 ;}; // on fait tout suivre
    forward only;
};

zone "." IN {
    type hint;
    file "named.ca"; // cache des racines
};

zone "1.0.0.127.in-addr.arpa" IN {
    type master;
    file "named.loopback"; // zone primaire du reverse loopback
};

zone "localhost" IN {
    type master;
    file "named.localhost";
};

```

**forward only**  
Ici, le S.esclave ne pourra jamais réinitier la résolution de la requête. Seuls, les S.forward seront utilisés. En cas de non réponse, la requête échouera.

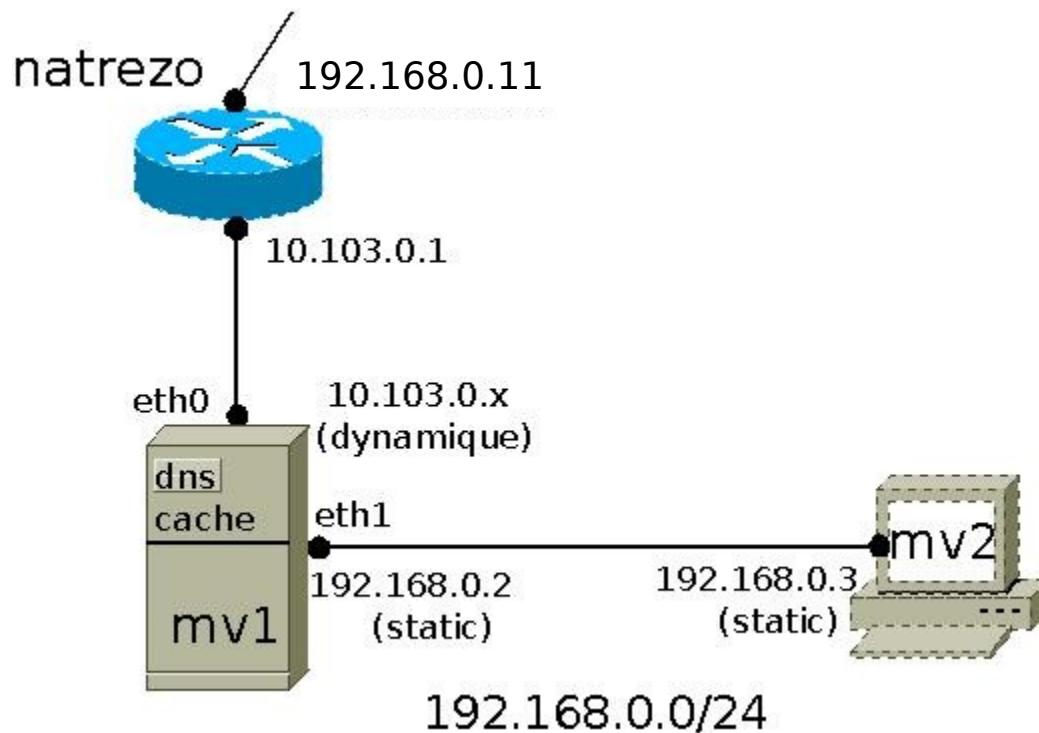
**forward first**  
La requête est d'abord envoyée aux S. forward. En cas de non réponse, le S.esclave se charge de réinitier la résolution de la requête lui-même.



# Exercices: Serveur de cache

## Exercice 1: Serveur de cache

- Réalisez la maquette suivante:



# Exercices: Serveur de cache

## Exercice 1: Serveur de cache (suite)

- Installez les packages de Bind sur MV1.
- Configurez le fichier **/etc/named.conf** sur MV1.
- Configurez les resolvers des 2 machines.

Sur MV1:

**/etc/resolv.conf**  
nameserver 127.0.0.1

Sur MV2:

**/etc/resolv.conf**  
nameserver 192.168.0.2

Liste des serveurs à contacter pour résoudre un nom - max 3.

- Vérifiez le fichier **/etc/nsswitch.conf** des 2 machines.

Sur MV1 & MV2

**/etc/nsswitch.conf**  
...  
hosts : files dns ...



# Exercices: Serveur de cache

## Exercice 1: Serveur de cache (suite)

- Vérifiez le fichier `/etc/hosts` des 2 machines.

Sur MV1 & MV2

/etc/hosts

127.0.0.1 localhost

- Lancez votre dns + vérification des logs dans `/var/log/messages`  
+ vérification via `named-checkconf` et `named-checkzone`
- Vérifiez le bon fonctionnement de votre dns à l'aide de nslookup et dig.  
(Vitez le cache entre vos manipulations nslookup et dig ...)
- Tentez de résoudre une requête dns à partir de MV2.
- Lancez wireshark sur MV2 et espionnez une requête dns.
- Lancez wireshark sur MV1 et espionnez une requête dns.
  - a) qui ne se trouve pas encore en cache
  - b) qui se trouve déjà en cache



# Exercices: Serveur de cache

## Exercice 2: Serveur forward esclave

- Modifiez la configuration du serveur de cache de l'exercice 1 pour que celui-ci devienne un serveur de type forward esclave.

Les serveurs forward à contacter :

- . celui de l'institut
- . ceux du FAI de l'institut

- Lancez votre dns + vérification des logs dans /var/log/messages  
+ vérification via named-checkconf et named-checkzone
- Vérifiez le bon fonctionnement de votre dns à l'aide de nslookup et dig.
- Tentez de résoudre une requête dns à partir de MV2.
- Lancez wireshark sur MV2 et espionnez une requête dns.
- Lancez wireshark sur MV1 et espionnez une requête dns lancée de MV2
  - a) qui ne se trouve pas encore dans le cache de MV1
  - b) qui se trouve déjà dans le cache de MV1



# Serveur autoritaire

## Serveur primaire

- Le serveur autoritaire d'une zone est la source des informations relatives à cette zone (il possède la base de données maître).
- Il contient les informations à partir d'un fichier de données où l'on effectue les mises à jour.
- Il a l'origine de l'autorité (Start Of Authority - SOA) sur cette zone.
- Il peut avoir autorité sur un ou plusieurs fichiers de zone.



# Serveur autoritaire

## Serveur secondaire

- C'est un miroir sauvegardé sur disque de la base de données maître.
  - Fonction de sauvegarde.
  - Fonction de répartition de charge et d'optimisation de l'accès suivant la situation géographique par rapport aux serveurs.
- Le serveur secondaire d'une zone obtient les informations relatives à celle-ci automatiquement depuis un serveur primaire ou un autre secondaire.
- Il a également autorité sur cette zone.
- Il peut avoir autorité sur un ou plusieurs fichiers de zone.



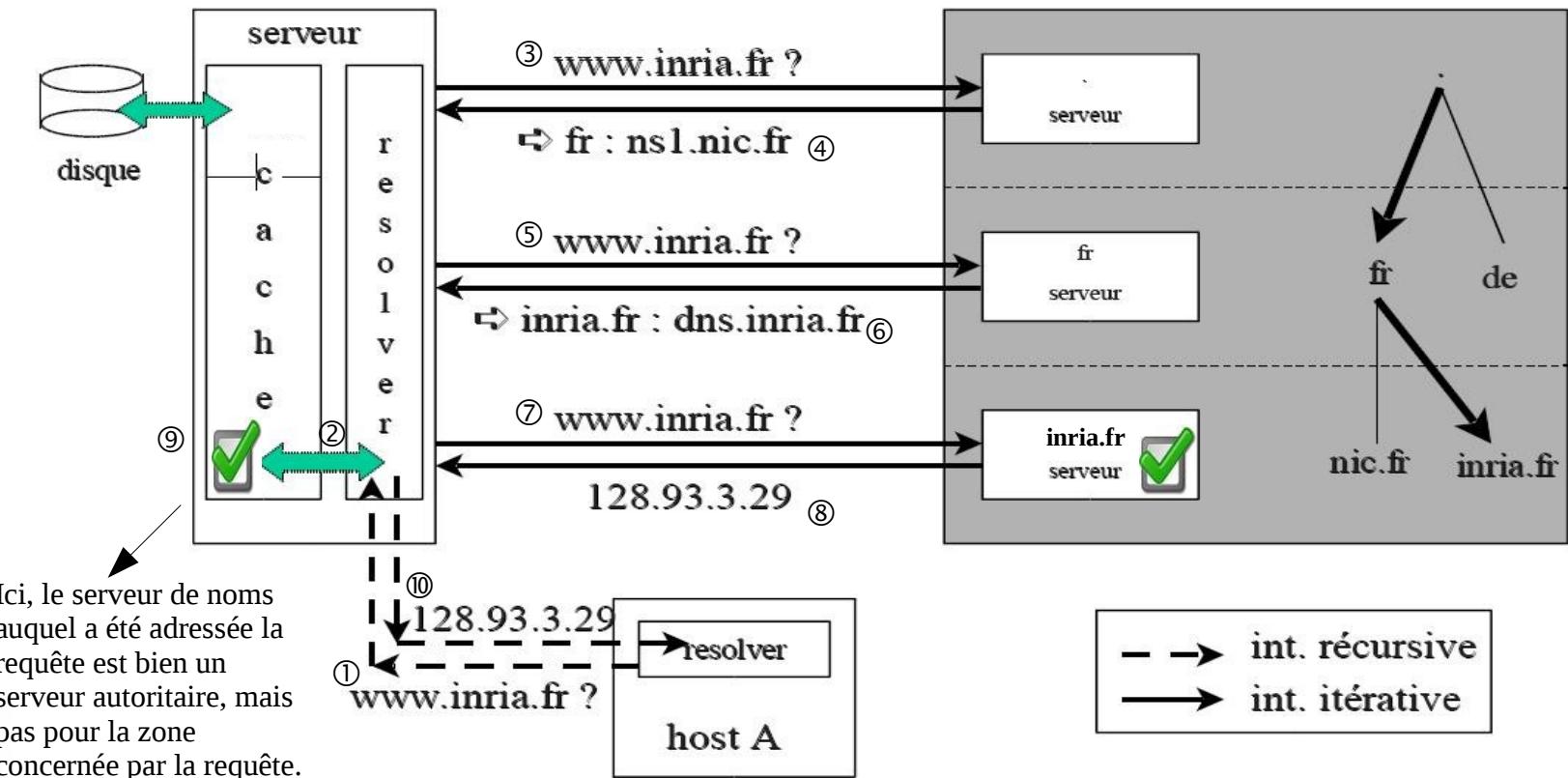
# Serveur autoritaire

## Architecture

Résolution d'une requête récursive par une suite de requêtes itératives envoyées vers les serveurs autoritaires des zones 'root' ('.'), 'fr' et 'inria.fr'.

Serveur ayant autorité sur le domaine afnic.fr

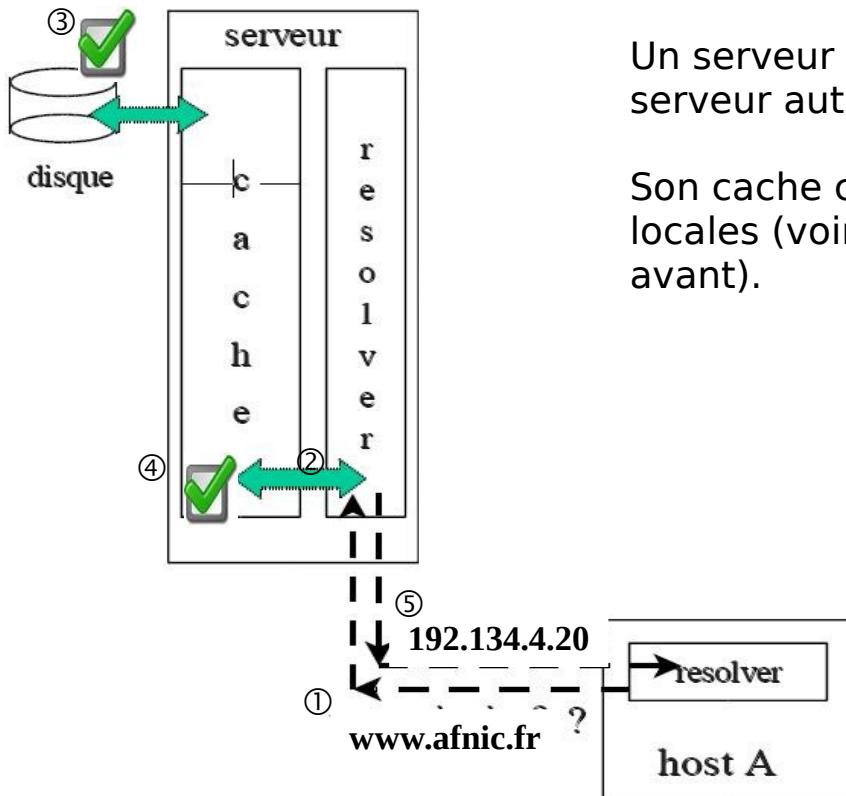
*Source: afnic - formation*



# Serveur autoritaire

## Architecture (suite)

Résolution d'une requête récursive envoyée directement vers le serveur autoritaire la zone concernée.



Un serveur peut être à la fois serveur cache et serveur autoritaire pour des zones.

Son cache contient alors aussi bien des données locales (voir ci-contre) que non locales (voir ci-avant).

—→ int. récursive  
—→ int. itérative

*Source: afnic - formation*



# Serveur autoritaire

## Remarques

### Récursif

- le serveur résout les requêtes récursives des clients et garde les informations obtenues dans son cache
  - le cache stocke des informations pour lesquelles le serveur n'a pas nécessairement autorité
- serveurs cache de campus par exemple

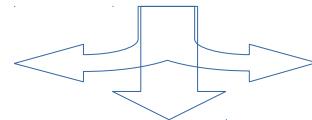
### Itératif

- il répond toujours en fonction des données qu'il possède localement
  - ne construit pas de cache pour des données non locales .
  - une machine cliente (d'utilisateur final) ne doit jamais pointer sur un serveur de ce type comme serveur par défaut.
- mode permettant de limiter la charge d'un serveur (il ne résout pas toute la requête)
  - serveurs de la racine, serveurs ayant autorité pour un grand nombre de zones.

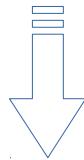


# Base de données

## Base de données

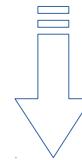


### Fichier de configuration



- named.conf
- localisation du fichier serveurs de la racine, des fichiers de zone
- déclaration de l'autorité sur des zones (primaire / secondaire)
- mode de fonctionnement général récursif: *recursion yes*  
itératif: *recursion no*  
faisant suivre : *forward first/only*  
type : *master / slave*
- ...

### Fichier des serveurs de la racine



- départ d'une résolution d 'une requête si le serveur n'a aucune information
- contient la liste des serveurs de nom et leur adresse IP
- primaire : A.ROOT-SERVERS.NET.
- 13 serveurs à travers le monde
- peut être généré de cette façon:  
dig @a.root-servers.net . ns > root.cache ou téléchargeable via :  
<ftp://ftp.rs.internic.net/domain>

### Fichier de zone



- contient les données propres à chaque zone
- consiste en une liste de Ressource Records (RR)
- le premier RR doit être le SOA (Start of Authority) qui décrit l'autorité technique de la zone
- directives spéciales: (\$ORIGIN, \$INCLUDE, \$TTL)

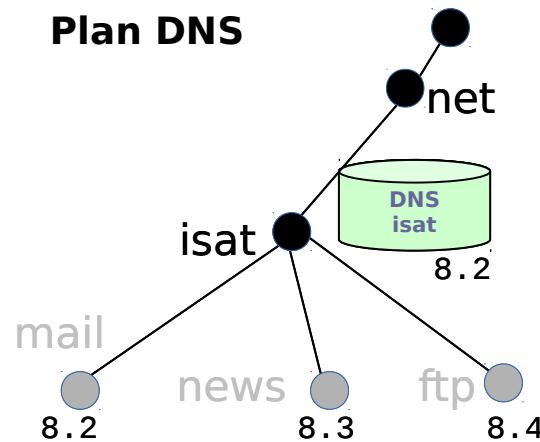


# Bind

## Gérer un domaine

Soit construire un serveur dns gérant le domaine 'isat.net' suivant:

### Plan DNS



Chaque machine devra être capable de répondre à une requête via son nom dns.

ex. ping ftp.isat.net devra fonctionner au sein de votre réseau 192.168.8.0/24

Chaque machine devra toujours être capable d'utiliser l'internet.

### Les services

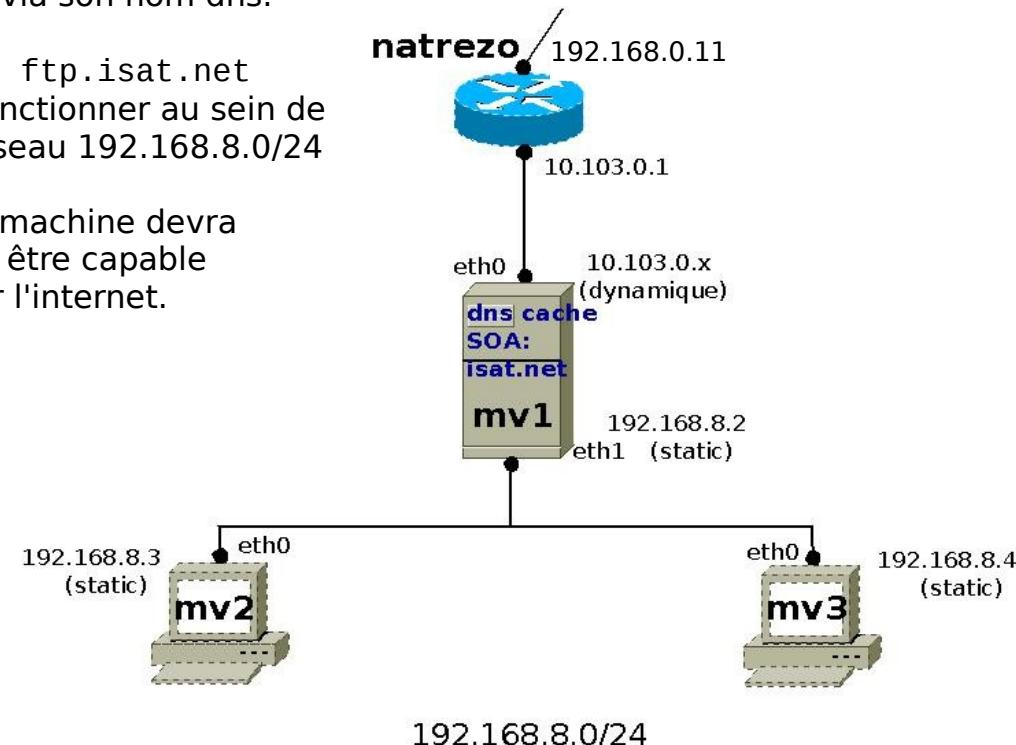
mail.isat.net sur 192.168.8.2  
news.isat.net sur 192.168.8.3  
ftp.isat.net sur 192.168.8.4

### Alias sur les noms canoniques

r2d2.isat.net sur 192.168.8.2  
yoda.isat.net sur 192.168.8.3  
lea.isat.net sur 192.168.8.4

### Le DNS

De la zone isat.net sur 192.168.8.2 Dns (JL Gouwy)



# Bind

## Serveur autoritaire récursif

### named.conf

```
options {
    listen-on port 53 {127.0.0.1 ; 192.168.8.2 ;} ; // Port d'écoute, ip admises
    directory "/var/named";
    recursion yes;                                //récursif (yes par défaut)
};
```

```
zone "." IN {
    type hint;
    file "named.ca";
};
```

```
zone "1.0.0.127.in-addr.arpa" IN {
    type master;
    file "named.loopback";
};
```

```
zone "localhost" IN {
    type master;
    file "named.localhost";
};
```

→ (Suite)

(Suite)

```
zone "isat.net" IN {
    type master;
    file "db.isat.net";
}; // zone primaire isat.net
```

```
zone "8.168.192.in-addr.arpa" IN {
    type master;
    file "db.isat.net-rev";
}; // zone primaire du reverse isat.net
```



# Bind

## Serveur autoritaire itératif

### named.conf

```
options {  
    listen-on port 53 {127.0.0.1 ; 192.168.8.2 ; } ; // Port d'écoute, ip admises  
    directory "/var/named";  
    recursion no; // le serveur n'accepte aucune requête récursive mais uniquement les  
    // requêtes itératives  
};
```

...  
idem ci-avant  
...



# Bind

## Syntaxe d'un RR:

**nom|@ [TTL] [classe] type données [commentaire]**

*Nom DNS auquel la ressource considérée est associée. Dans un fichier de zone, les noms se terminant par "." sont des FQDN tandis que les autres sont relatifs au nom de la zone.*

*@ : utilise la valeur courante de \$ORIGIN comme nom (ou le nom de zone indiquée dans named.conf).*

*Nombre spécifiant la durée pendant laquelle le RR peut être conservé en cache. Si aucune valeur n'est indiquée, c'est le TTL par défaut de la zone qui est appliquée (en sec. Ou M-H-D-W).*

*Type de l'enregistrement (A - NS - MX - SOA ...).*

*Internet (pour des raisons pratiques, c'est la seule classe importante). Si elle est absente, c'est IN par défaut qui est employée.*

*Données spécifiques au type d'enregistrement.*



# Bind

**Les fichiers de zone:** celui de la zone isat.net

```
$ORIGIN isat.net.
$TTL 2D
isat.net.    IN      SOA     ns.isat.net.  root.isat.net.  (
                                         2012110700 ; Serial
                                         28800       ; Refresh
                                         14400       ; Retry
                                         3600000    ; Expire
                                         7200 )      ; Minimum

                                         IN      NS      ns.isat.net.

ns          IN      A       192.168.8.2
mail        IN      A       192.168.8.2
news        IN      A       192.168.8.3
ftp         IN      A       192.168.8.4

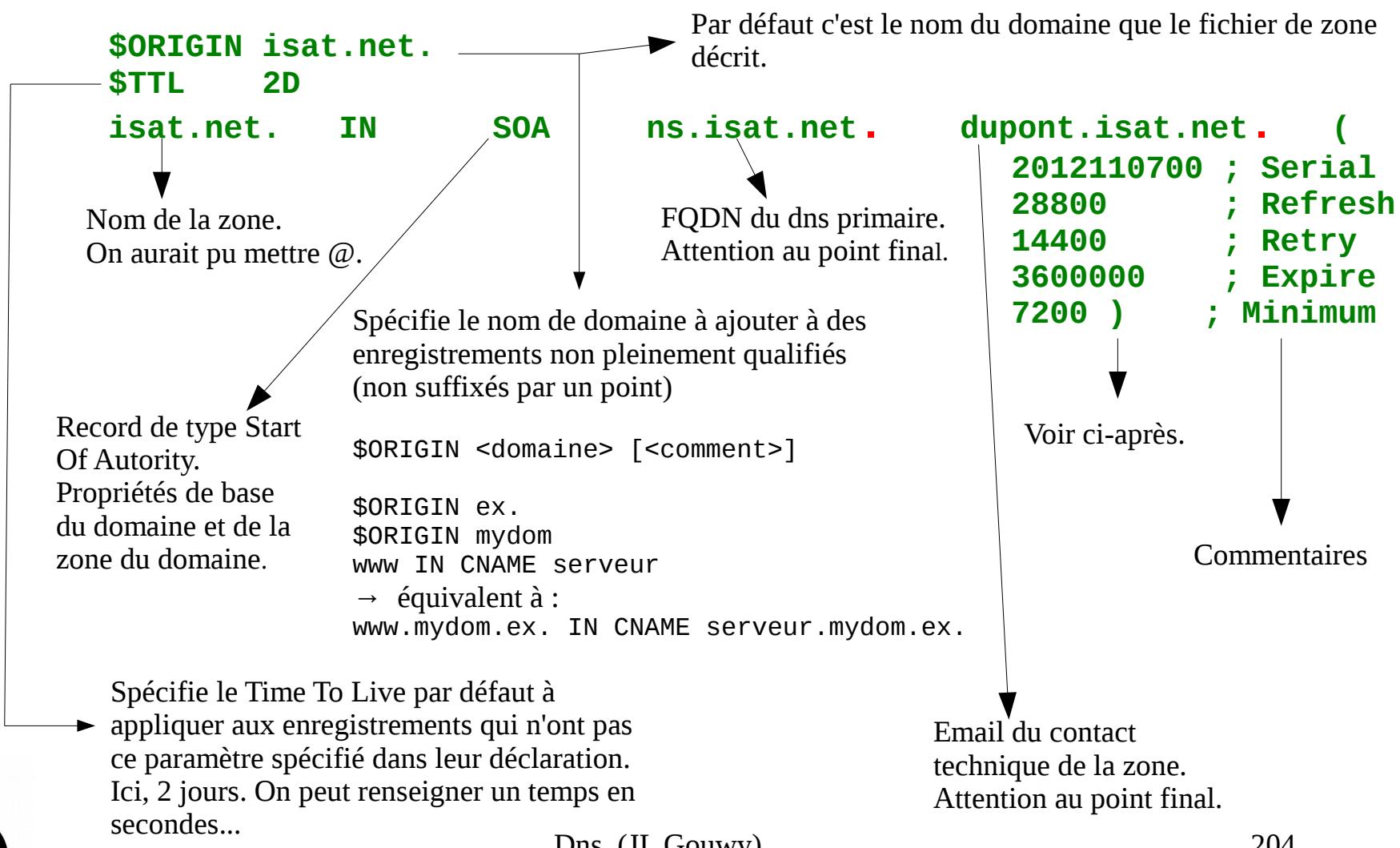
r2d2        IN      CNAME   mail
lea         IN      CNAME   ftp
yoda        IN      CNAME   news

isat.net.    IN      MX      10      mail
```



# Bind

## Les fichiers de zone: celui de la zone isat.net (suite)



# Bind

**Les fichiers de zone:** celui de la zone isat.net (suite)

```
$ORIGIN isat.net.
$TTL 2D
isat.net. IN SOA ns.isat.net. dupont.isat.net. (
    2012110700 ; Serial
    28800       ; Refresh
    14400       ; Retry
    3600000    ; Expire
    7200 )      ; Minimum
```

Serial: Spécifie la version des données de la zone.  
 A incrémenter à chaque modification (nécessaire pour la synchronisation des serveurs secondaires)  
 Conseil : YYYYMMDDxx → max. 99 modif./jour

Refresh: Intervalle, ici en sec., entre 2 vérifications du serial number par les secondaires.

Retry: Intervalle, ici en sec., entre 2 vérifications du serial number par les secondaires si la 1ere vérification a échoué.

Expire: Temps, ici en sec., après lequel le secondaire détruit les données de la zone qu'il possède et arrête de répondre aux requêtes pour cette zone s'il ne parvient pas à contacter le serveur primaire.

**retry<<refresh<<expire**

Minimum:

Temps que doit rester dans le cache une réponse négative suite à une question sur ce domaine.

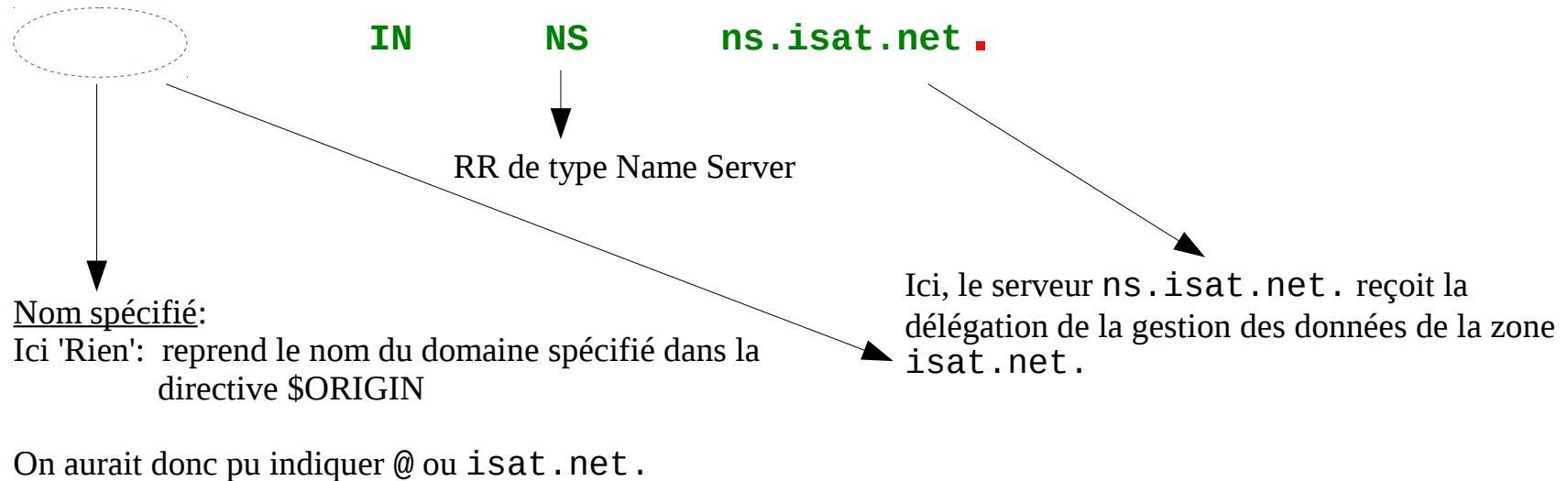
Deux types de réponses négatives :

- NXDOMAIN : aucun RR ayant le nom demandé dans la classe (IN) n'existe dans cette zone.
- NODATA : aucune donnée pour le triplet (nom, type, classe) demandé n'existe ; il existe d'autres records possédant ce nom, mais de type différent.



# Bind

**Les fichiers de zone:** celui de la zone isat.net (suite)



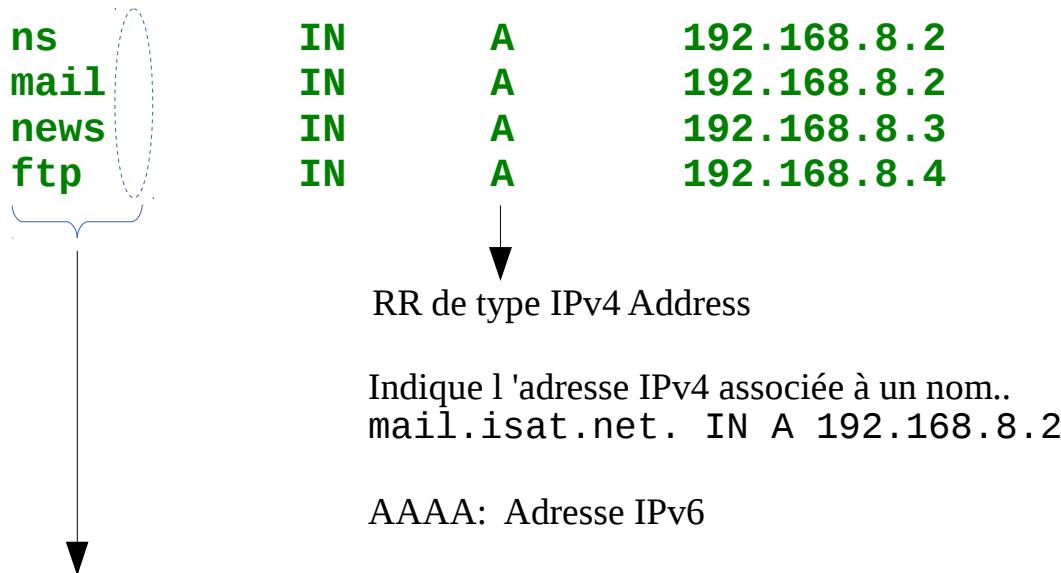
zone	IN	NS	serveur-nom1.domaine.	→ serveur secondaire
	IN	NS	serveur-nom2.domaine.	

Il faut spécifier les serveurs de noms de la zone que l'on décrit (associé au SOA)



# Bind

**Les fichiers de zone:** celui de la zone isat.net (suite)



Les noms non pleinement qualifiés (ne se terminant pas par un point) sont relatifs au nom du domaine spécifié dans la directive \$ORIGIN  
Donc, ici, les écritures :

mail.isat.net. IN A 192.168.8.2  
et  
mail IN A 192.168.8.2

sont équivalentes.



# Bind

**Les fichiers de zone:** celui de la zone isat.net (suite)

r2d2	IN	CNAME	mail
lea	IN	CNAME	ftp
yoda	IN	CNAME	news

RR de type 'Canonical name'

Indique que le nom est un alias  
vers un autre nom (le nom canonique)

alias IN CNAME nom.canonique.

Remarques:

- plusieurs alias différents peuvent pointer vers le même nom canonique

alias1 IN CNAME relais  
alias2 IN CNAME relais



☺ si l'Ip associée au nom canonique change, seule l'Ip correspondant à ce nom canonique dans son RR de type A doit être changée.

- un nom canonique ne doit pas pointer vers un alias déjà créé

alias IN CNAME relais1  
alias IN CNAME relais2 →

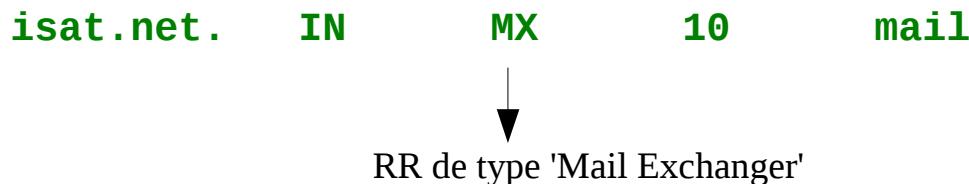
- quand un nom est déjà lié à un CNAME, il est interdit de faire figurer d'autres types de RR pour ce nom.

alias IN CNAME relais  
alias IN NS serveur →



# Bind

**Les fichiers de zone:** celui de la zone isat.net (suite)



Spécifie un serveur de messagerie pour la zone : email à quelqu - un@nom

On cherche dans le DNS un MX indiquant la machine sur laquelle il faut envoyer le courrier pour nom.

Un paramètre précise le poids relatif de l'enregistrement MX:

Si plusieurs MX existent, le courrier est envoyé en 1er à la machine ayant le poids le plus bas, puis dans l'ordre croissant des poids en cas d'échec

```
nom      IN MX 10 nom.relais1.  
          IN MX 20 nom.relais2.  
          IN MX 30 nom.relais3.
```



# Bind

**Les fichiers de zone:** celui de la zone reverse 8.168.192.in-addr.arpa

```
$ORIGIN 8.168.192.in-addr.arpa.
$TTL 2D
8.168.192.in-addr.arpa. IN SOA ns.isat.net. root.isat.net. (
                           2012110700 ; Serial
                           28800      ; Refresh
                           14400      ; Retry
                           3600000   ; Expire
                           7200       ; Minimum
```

IN NS ns.isat.net.

2	IN PTR mail.isat.net.
3	IN PTR news.isat.net.
4	IN PTR ftp.isat.net.



RR de type 'Pointeur'

Indique le nom associé à un numéro IP dans l'arborescence in-addr.arpa (ip6.arpa)

2.8.168.192.in-addr.arpa. IN PTR mail.isat.net.



# Bind

**Les fichiers de zone:** celui du reverse 0.0.127.in-addr.arpa

- Personne n'a la responsabilité de ce reverse pour le numéro 127.0.0.1 dans la hiérarchie in-addr.arpa.
- Doit toujours être configuré sous peine de comportement anormal du DNS.

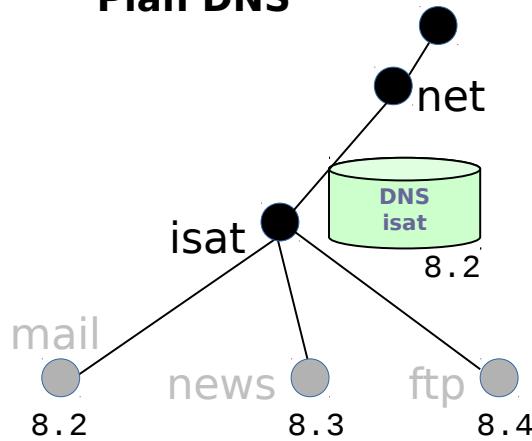


# Exercice: Serveur autoritaire

## Exercice 3: Serveur autoritaire de cache

- Construire le serveur dns gérant le domaine 'isat.net.' exposé ci-avant.

### Plan DNS



Chaque machine devra être capable de répondre à une requête via son nom dns.

ex. ping ftp.isat.net  
devra fonctionner au sein de votre réseau 192.169.8.0/24

Chaque machine devra toujours être capable d'utiliser l'internet.

#### Les services

mail.isat.net sur 192.168.8.2  
news.isat.net sur 192.168.8.3  
ftp.isat.net sur 192.168.8.4

#### Alias sur les noms canoniques

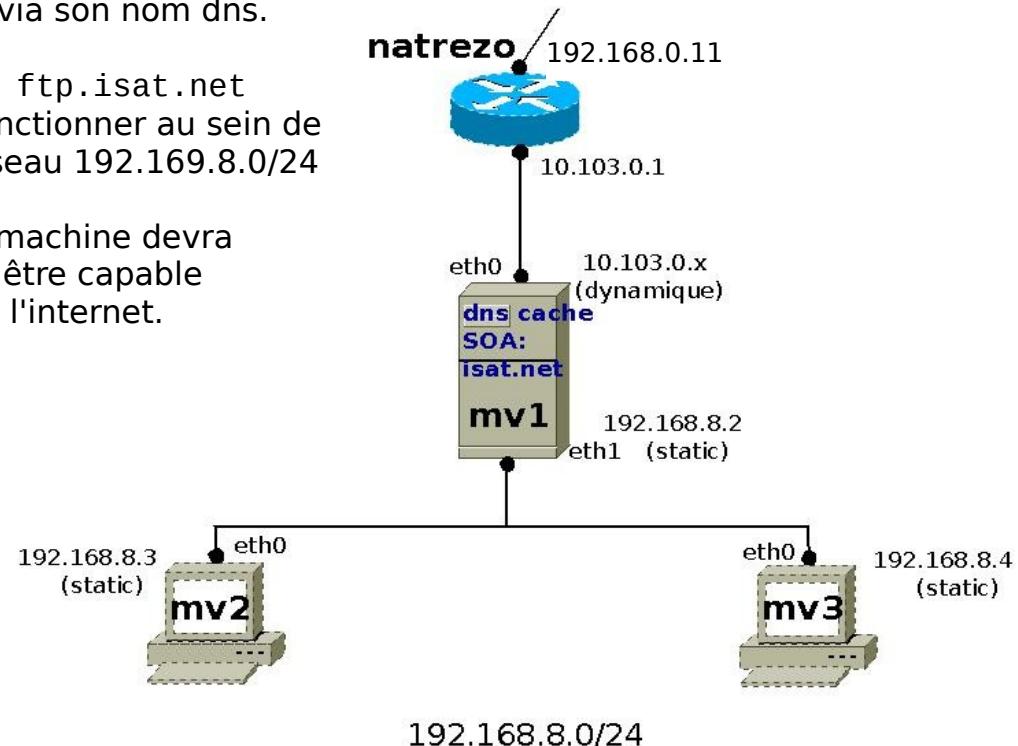
r2d2.isat.net sur 192.168.8.2  
yoda.isat.net sur 192.168.8.3  
lea.isat.net sur 192.168.8.4

#### Le DNS

De la zone "isat.net" sur 192.168.8.2



Dns (JL Gouwy)



# Exercice: Serveur autoritaire

## Exercice 3: Serveur autoritaire de cache

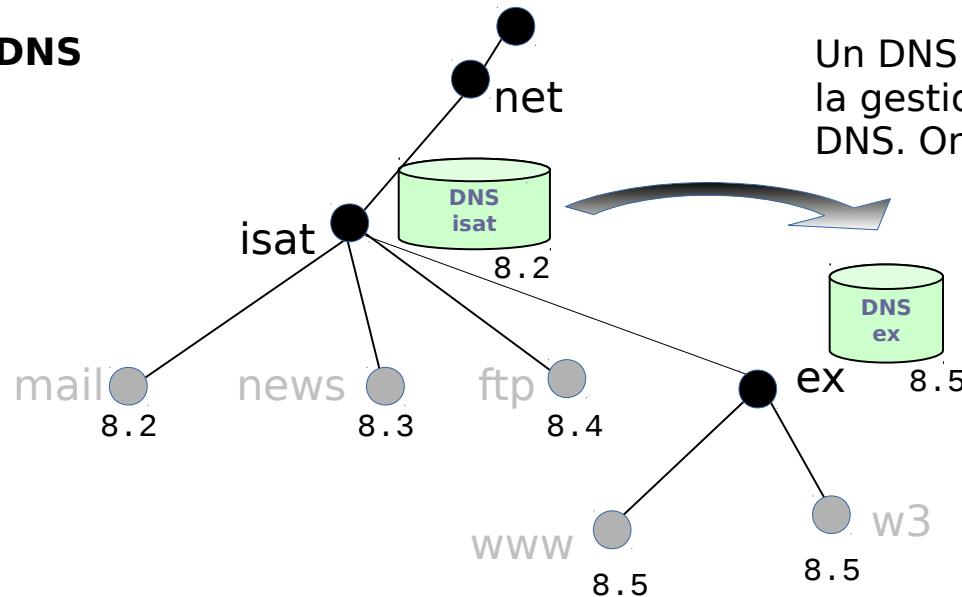
- Configurez votre dns.
- Fonctionnement (ping + outils de debugage):
  - . chaque mv doit pouvoir se toucher par son nom ou son alias.
  - . chaque mv a toujours l'accès à l'internet.
- Quelques essais et compléments
  - a) Enlevez 192.168.8.2. ; de la directive listen-on et relancez named . Que constatez-vous ?
  - b) Rajoutez à nouveau ce réseau et ajoutez la directive recursion no et relancez named . Que constatez-vous ?
  - c) Remplacez la directive recursion no par:  
*recursion yes → facultatif car par défaut*  
allow-recursion {127.0.0.1 ; 192.168.8.0/24;} ;  
allow-query-cache {127.0.0.1 ; 192.168.8.0/24;} ;  
et relancez named . Que constatez-vous ?
  - d) Rajoutez la directive version "DNS ISAT" et relancez named . Quelle pourrait-être son utilité ?



# Délégation et sous domaine

Soit une nouvelle zone 'ex':

## Plan DNS



Un DNS transfère la responsabilité de la gestion du sous domaine à un autre DNS. On dit qu'il délègue son autorité.

### Les services

mail.isat.net sur 192.168.8.2  
news.isat.net sur 192.168.8.3  
ftp.isat.net sur 192.168.8.4

www.ex.isat.net sur 192.168.8.5  
w3.ex.isat.net sur 192.168.8.5

### Les DNS

De la zone "isat.net" sur 192.168.8.2  
De la zone "ex.isat.net" sur 192.168.8.5



# Délégation et sous domaine

## Bind: Le fichier de la zone parente

```
$ORIGIN isat.net.
$TTL 2D
isat.net.      IN      SOA      ns.isat.net.  root.isat.net.  (
                                         2012110700 ; Serial
                                         28800      ; Refresh
                                         14400      ; Retry
                                         3600000   ; Expire
                                         7200 )    ; Minimum
                                         IN      NS      ns.isat.net.
ex            IN      NS      ns.ex.isat.net.
                                         IN      A       192.168.8.2
ns             IN      A       192.168.8.2
mail           IN      A       192.168.8.3
news           IN      A       192.168.8.4
ftp            IN      A       192.168.8.5
ns.ex          IN      A       192.168.8.5
                                         IN      MX     10      mail
```

La délégation de zone est déclarée dans le fichier de zone du domaine parent par un RR de type NS.

Et un RR de type A est ensuite nécessaire pour la correspondance entre l'adresse IP et le nom.



# Délégation et sous domaine

## Bind: Le fichier de la zone fille

```
$ORIGIN ex.isat.net.  
$TTL 2D  
ex.isat.net. IN SOA ns.ex.isat.net. root.ex.isat.net. ( 2012110700 ; Serial  
28800 ; Refresh  
14400 ; Retry  
3600000 ; Expire  
7200 ) ; Minimum  
IN NS ns.ex.isat.net.  
  
ns IN A 192.168.8.5  
www IN A 192.168.8.5  
w3 IN A 192.168.8.5
```



Le fichier de zone du sous-domaine est un fichier de zone classique.

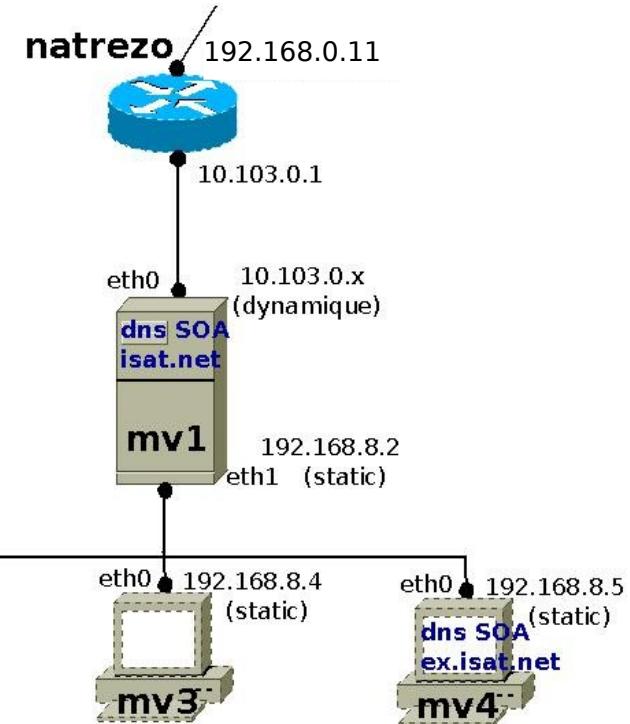
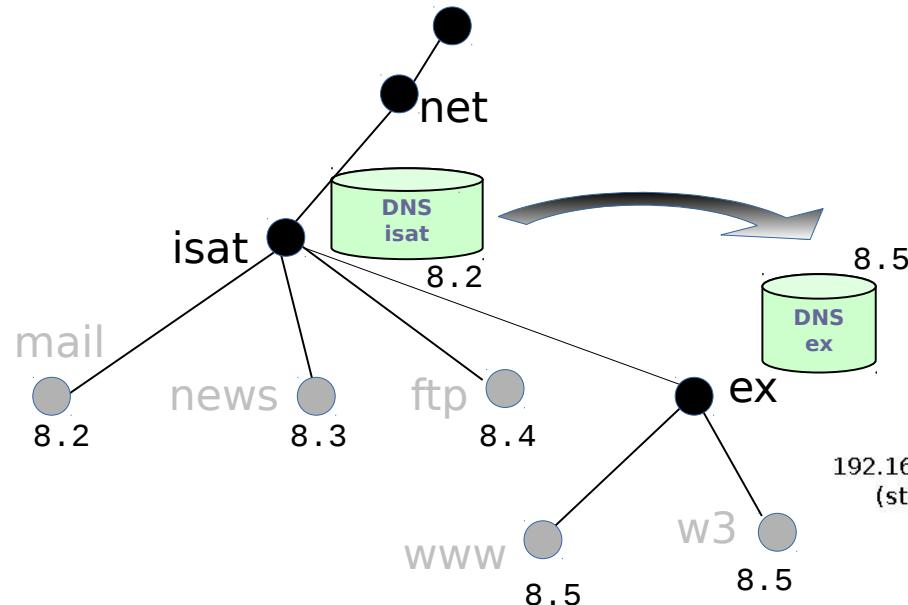


# Exercice: Délégation et sous domaine

## Exercice 4

- Construire les serveurs dns gérant les domaines isat.net. et ex.isat.net :

Plan DNS et architecture:



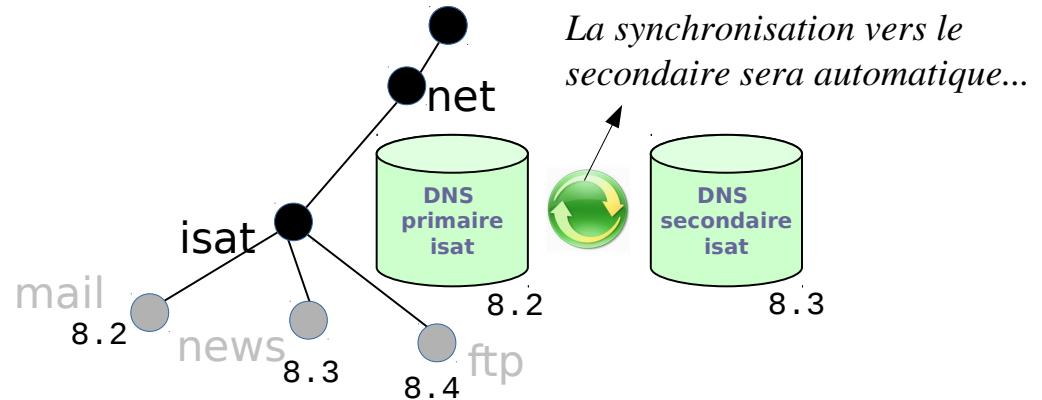
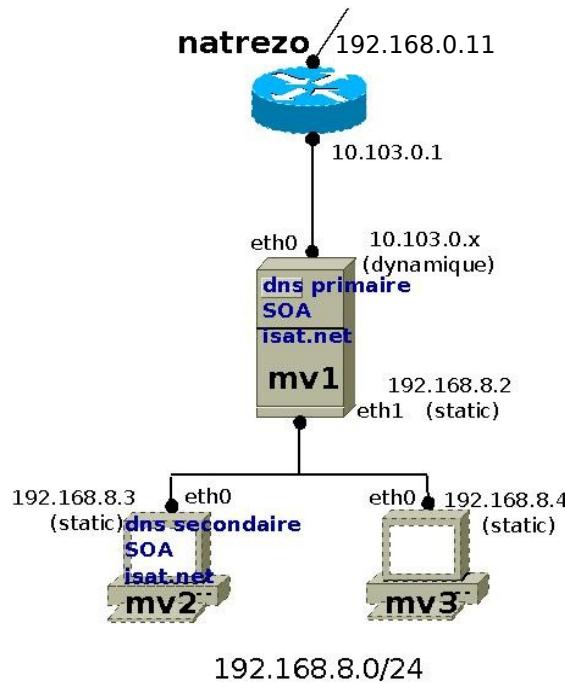
- Fonctionnement (ping + outils de debugage):
  - . chaque mv doit pouvoir se toucher par son nom ou son alias.
  - . chaque mv a toujours l'accès à l'internet.

192.168.8.0/24



# Redondance

Soit ajouter un serveur dns secondaire pour le domaine 'isat.net':



- Un dns secondaire ne gère pas directement les informations sur les zones mais les obtient à partir du primaire de la zone (ou d'un autre secondaire) via le réseau (transfert de zone).
- Ce dns ne peut modifier des données de la zone mais il a lui aussi autorité sur la zone.
- Cette redondance permet une meilleure tolérance aux pannes et une réduction de la charge de travail des dns principaux.

2 façons pour mettre à jour un secondaire:

- en fonction de la valeur 'refresh' définie dans le SOA de la zone ;
- ou lorsqu'il reçoit une notification du primaire

Quand il démarre, un secondaire doit connaître son maître pour entamer un transfert de zone avec lui.



# Bind

Configuration du serveur primaire: **named.conf**

```
options {
    listen-on port 53 {127.0.0.1 ; 192.168.8.2 ;} ;
    directory "/var/named";
};
```

```
zone "." IN {
    type hint;
    file "named.ca";
```

*Ces esclaves ont le droit de demander le transfert de ces zones au maître.*

```
zone "1.0.0.127.in-addr.arpa" IN {
    type master;
    file "named.loopback";
```

```
zone "localhost" IN {
    type master;
    file "named.localhost";
```

*(Suite)*

*Le maître a le devoir de signaler les changements sur ces zones aux esclaves.*

*(Suite)*

```
zone "isat.net" IN {
    type master;
    notify yes;
    also-notify {192.168.8.3;} ;
    allow-transfer {192.168.8.3;} ;
    file "db.isat.net";
}; // zone primaire isat.net
```

```
zone "8.168.192.in-addr.arpa" IN {
    type master;
    notify yes;
    also-notify {192.168.8.3;} ;
    allow-transfer {192.168.8.3;} ;
    file "db.isat.net-rev";
}; // zone primaire du reverse isat.net
```



# Bind

Configuration du serveur primaire: **db.isat.net**

```
$ORIGIN isat.net.
$TTL    2D
isat.net.  IN  SOA      ns.isat.net.  root.isat.net. (
2012110701 ; Serial
28800       ; Refresh
14400       ; Retry
3600000    ; Expire
7200        ; Minimum
```

On "déclare" ns2.isat.net comme serveur secondaire.  
Attention, le RR du serveur primaire (ns.isat.net) doit rester en première position (voir après).

	IN	NS	ns.isat.net.
	<b>IN</b>	<b>NS</b>	<b>ns2.isat.net.</b>

ns	IN	A	192.168.8.2
<b>ns2</b>	<b>IN</b>	<b>A</b>	<b>192.168.8.3</b>
mail	IN	A	192.168.8.2
news	IN	A	192.168.8.3
ftp	IN	A	192.168.8.4

r2d2	IN	CNAME	mail
lea	IN	CNAME	ftp
yoda	IN	CNAME	news

isat.net.	IN	MX	10	mail
-----------	----	----	----	------

Chaque fois qu'une zone du maître est modifiée, ne pas oublier d'incrémenter le numéro de série (utilisé pour la synchronisation)



# Bind

Configuration du serveur primaire: **db.isat.net-rev**

```
$ORIGIN 8.168.192.in-addr.arpa.
$TTL 2D
8.168.192.in-addr.arpa. IN SOA ns.isat.net. root.isat.net. (
                           2012110701 ; Serial
                           28800    ; Refresh
                           14400    ; Retry
                           3600000 ; Expire
                           7200     ; Minimum
                           )
3      IN      NS      ns.isat.net.
3      IN      NS      ns2.isat.net.
2      IN      PTR     mail.isat.net.
3      IN      PTR     news.isat.net.
4      IN      PTR     ftp.isat.net.
```

*On "déclare" également le serveur secondaire dans la zone reverse sans oublier d'incrémenter le numéro de série.*



# Bind

## Configuration du serveur secondaire: **named.conf**

```
options {
    listen-on port 53 {127.0.0.1 ; 192.168.8.3 ;} ;
    directory "/var/named";
};
```

```
zone "." IN {
    type hint;
    file "named.ca";
};
```

```
zone "1.0.0.127.in-addr.arpa" IN {
    type master;
    file "named.loopback";
};
```

```
zone "localhost" IN {
    type master;
    file "named.localhost";
};
```

(Suite)

*Les zones transférées seront stockées dans /var/named/slaves.  
Le process named devra avoir la permission d'y écrire.  
Ce dossier sera garni par le serveur lui-même.*

(Suite)

```
zone "isat.net" IN {
    type slave;
    masters { 192.168.8.2; } ;
    file "slaves/db.isat.net";
}; // zone secondaire isat.net
```

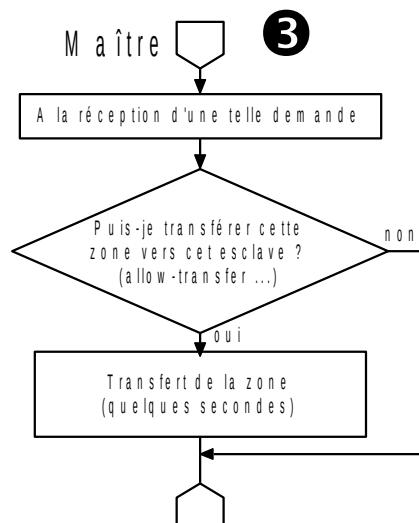
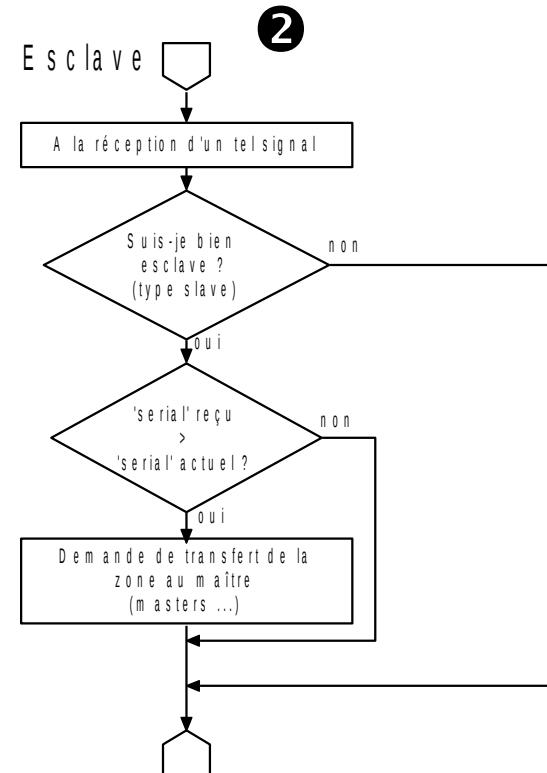
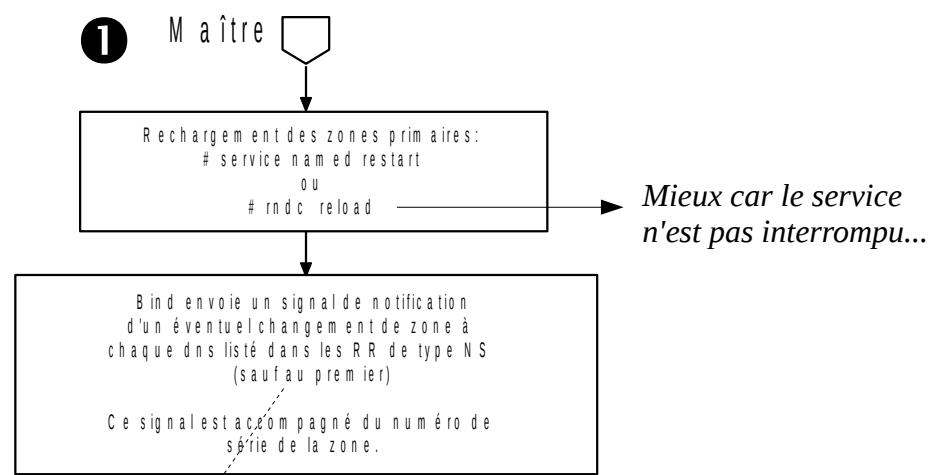
```
zone "8.168.192.in-addr.arpa" IN {
    type slave;
    masters { 192.168.8.2; } ;
    file "slaves/db.isat.net-rev";
}; // zone secondaire du reverse isat.net
```

*Ces zones dites "non-vivantes" ne risquent pas changer.  
ns2 peut alors en être maître. Elles sont identiques à celles du primaire.*



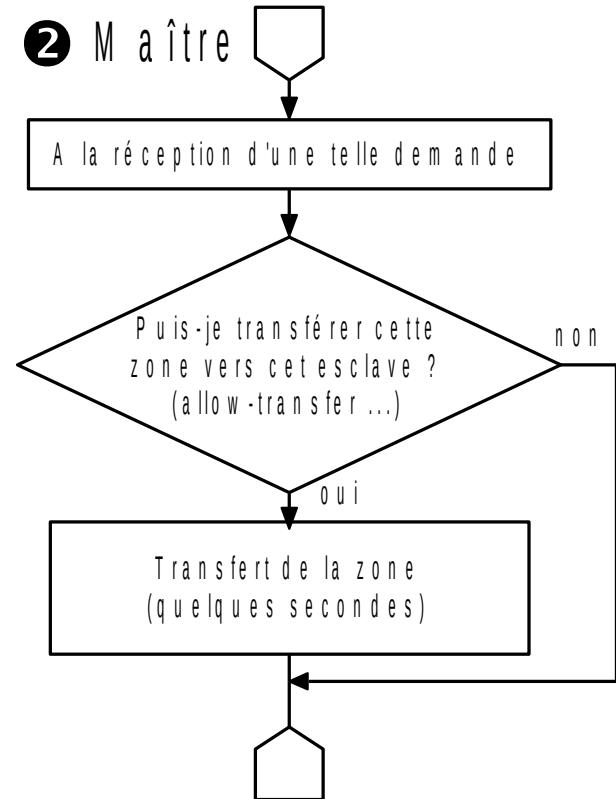
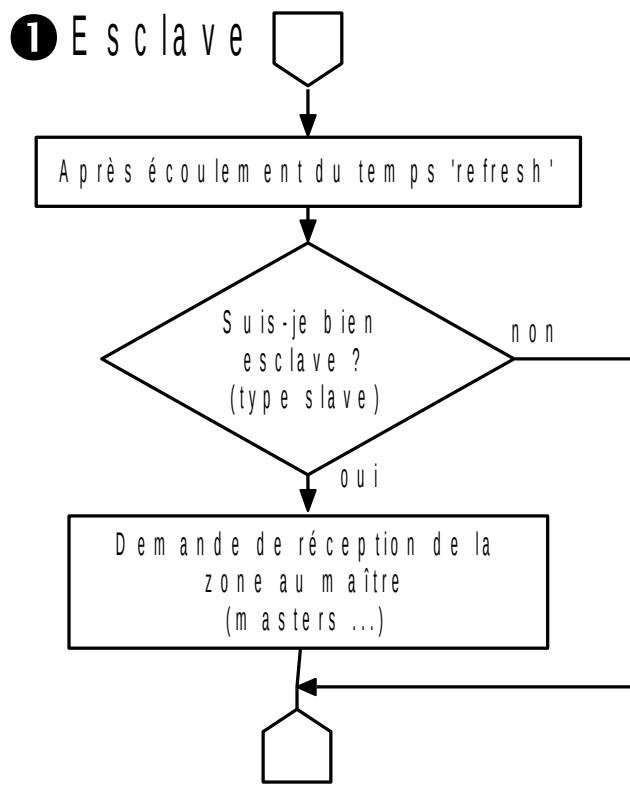
# Bind

## Synchronisation (1): Le maître notifie



# Bind

## Synchronisation (2): Le maître ne notify pas



- ☞ Ici, et par souci de clarté, les temps 'retry' et 'expire' sont volontairement écartés de l'explication ...
- ☞ Cette technique de synchronisation est de moins en moins utilisée car la synchronisation est trop tardive par rapport à la technique par 'notification'.

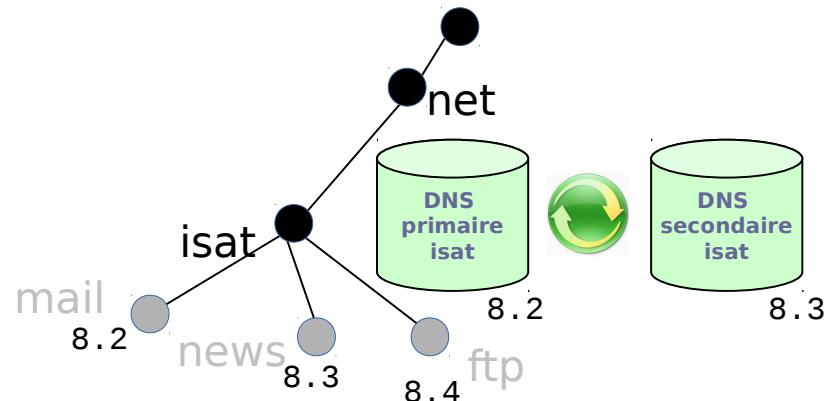
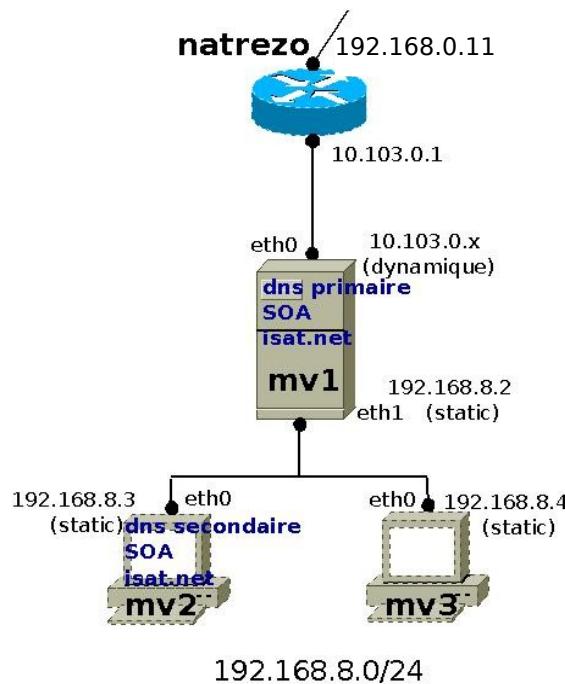


# Exercice: Redondance

## Exercice 5

- Construire les serveurs dns primaires et secondaires de 'isat.net':

Plan DNS et architecture:



### Les services

mail.isat.net sur 192.168.8.2  
news.isat.net sur 192.168.8.3  
ftp.isat.net sur 192.168.8.4

### Alias sur les noms canoniques

r2d2.isat.net sur 192.168.8.2  
yoda.isat.net sur 192.168.8.3  
lea.isat.net sur 192.168.8.4

### Les DNS

Primaire du domaine "isat.net" sur 192.168.8.2  
Secondaire du domaine "isat.net" sur 192.168.8.2



# Exercice: Redondance

## Exercice 5 (suite)

- Lancez un shell sur MV1 et MV2 pour surveiller l'évolution des logs.  
Sur un autre shell, démarrez bind sur le primaire puis sur le secondaire.  
Que constatez-vous ?
  
- Incrémentez le numéro de série et ajoutez un RR factice dans les 2 zones du primaire.  
Rechargez les zones du primaire.  
Que constatez-vous ?
  
- Sur MV3, déclarez MV1 et MV2 comme dns à contacter pour résoudre des noms et tentez une résolution de noms.  
Stoppez bind sur MV1.  
Retentez une résolution de noms à partir de MV3.  
Que constatez-vous ?



# Références

## WEBOGRAPHIE

<http://www.afnic.fr//ext/dns/index.html> (autoformation)  
<http://www.certa.ssi.gouv.fr/site/CERTA-2008-INF-002>  
[http://www.reseaucerta.org/docs/cotecours/ccDNS\\_v1.2.pdf](http://www.reseaucerta.org/docs/cotecours/ccDNS_v1.2.pdf)  
<http://www.centosadmin.net>  
<http://www.zytrax.com/books/dns>  
<http://zero202.free.fr/cs61-dns/html/ar01s03.html>

## BIBLIOGRAPHIE

LINUX - Préparation à la certification LPIC-1 (LPI 101 LPI 102) [2e édition]  
Par Sébastien ROHAUT - ENI Editions

CentOS Bible  
By Timothy Boronczyk and Christopher Negus  
Edition 2009 - Wiley Publishing, Inc.

ADMINISTRATION UNIX: ASPECTS RESEAUX  
Par Xavier Bogaert - Technofutur3





## Laboratoire de sécurité internet

# Découverte du protocole HTTP

Jean-Louis Gouwy



# Plan

- Le protocole HTTP
  - Introduction
  - Les versions
  - Interprétation d'une URL
  - Le modèle client/serveur
  - La requête client
  - La réponse serveur
- Ateliers de découverte
  - Mise en place
  - Atelier 1: Envoi d'une requête HTTP par telnet
  - Atelier 2: Envoi de cette requête en HTTP 1.0
  - Atelier 3: Codification d'une requête HTTP dans un script
  - Atelier 4: Requête vers une ressource qui n'existe pas
  - Atelier 5: Redirection d'une requête
  - Atelier 6: Accès à un site protégé
  - Atelier 7: Refus d'accès à une ressource
  - Atelier 8: Accès sécurisé à un site
  - Atelier 9: Les connexions persistantes
- Autres champs



# Plan

- Les méthodes des clients
  - Les méthodes classiques
  - La méthode GET
  - La méthode POST
  - GET vs POST
  - Ateliers d'espionnage
- Les codes de retour
- Référence



# Le protocole HTTP

- **IINTERPRETATION D'UNE URL (*Uniform Resource Locator*)**

Soit l'URL : **http://hypothetical.ora.com:80/**

Le navigateur interprète cette URL de la façon suivante :

**http://**

Utiliser HTTP comme protocole.

**hypothetical.ora.com**

Contacter un ordinateur sur le réseau portant ce nom.

**:80**

Se connecter sur son port 80. S'il est omis le navigateur le rajoute Automatiquement.

**/**

Chemin du document.



# Le protocole HTTP

- **INTRODUCTION**

- HyperText Transfert Protocol (proto. de transfert de document hypertexte)
- Inventé par Tim Berners-Lee avec les adresses Web et le langage HTML pour créer le World Wide Web.

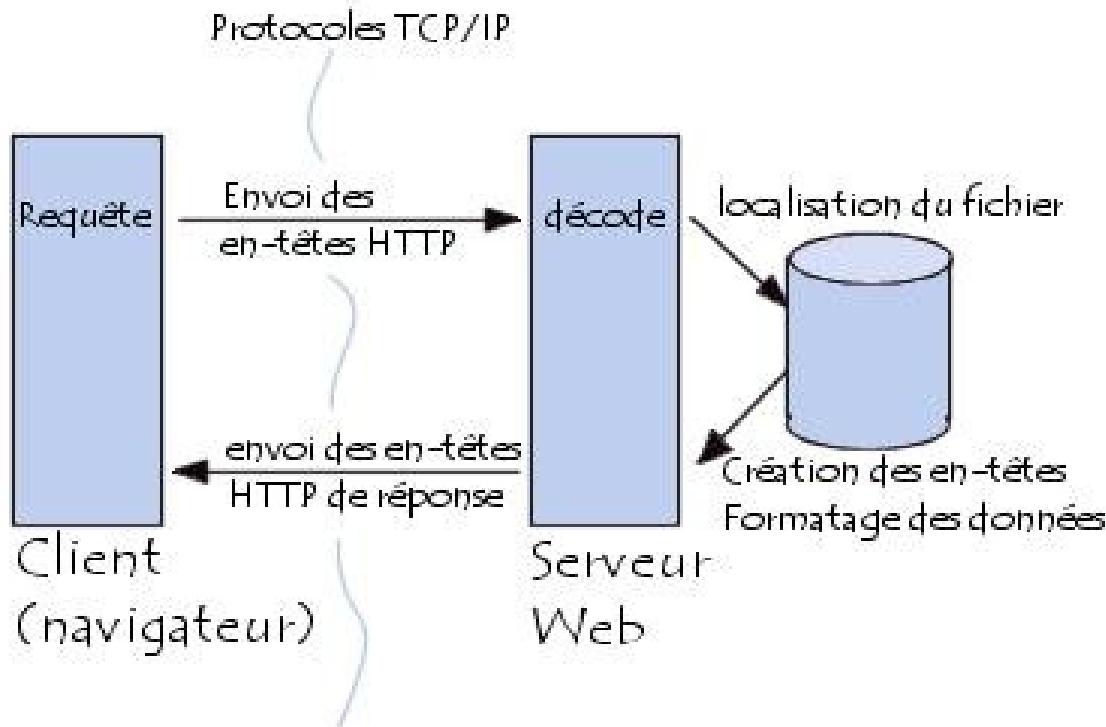
- **LES VERSIONS**

- 1990: HTTP/0.9
- 1996: HTTP/1.0 devient un standard. Cette version supporte les sites web virtuels, la gestion de cache, l'identification.
- 1997: HTTP/1.1 devient le nouveau standard . Cette version ajoute notamment les connexions persistantes, la négociation de type de contenu (format de données, langue), les sites web virtuels hébergés par Ip
- 2015: HTTP/2 est poussé par Google mais encore très controversé. Cette version se targue d'améliorer sensiblement les performances de chargement de pages lourdes. A suivre....



# Le protocole HTTP

- LE MODELE CLIENT/SERVEUR



Source: <http://www.commentcamarche.net/contents/internet/http.php3>



# Le protocole HTTP

- **LA REQUETE CLIENT**

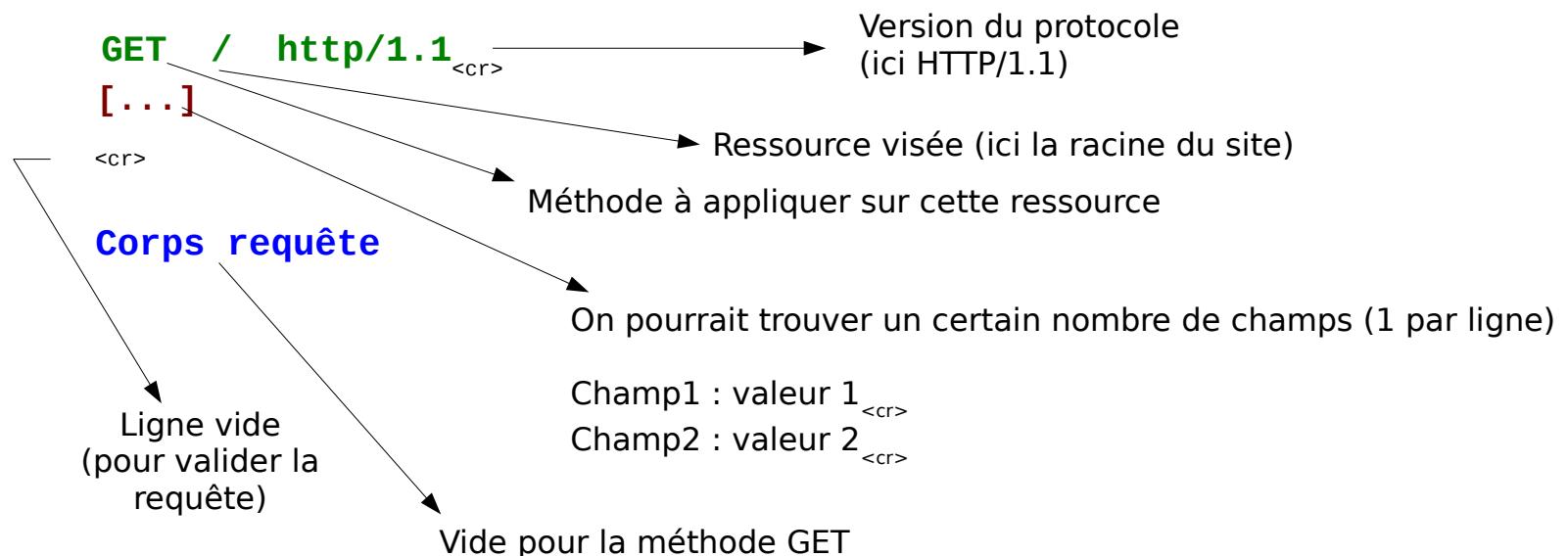
Syntaxe d'une requête client

**Ligne de commande (Commande, URL, Version de protocole)**

**En-tête de requête**

<cr>

**Corps de requête**



# Le protocole HTTP

- **LA REPONSE SERVEUR**

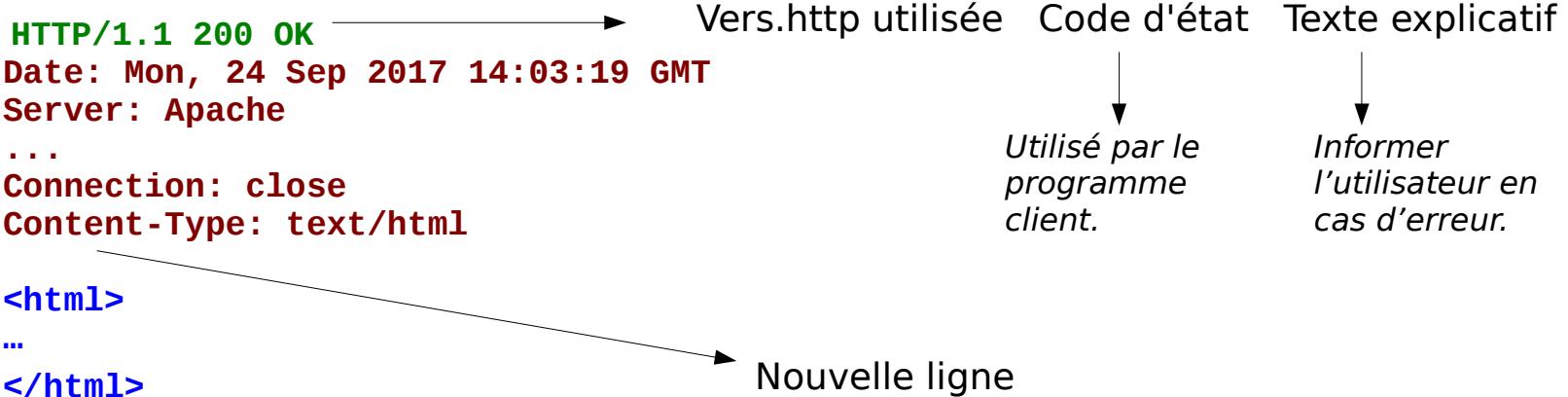
Syntaxe d'une réponse serveur

**Ligne de statut (Version, Code-réponse, Texte-réponse)**

**En-tête de réponse**

<nouvelle ligne>

**Corps de réponse**



# Ateliers de découverte

- **MISE EN PLACE**

- Pour ces ateliers, nous travaillerons avec 4 sites `www.test.be`, `bank.test.be`, `get.test.be` et `post.test.be` hébergés par un serveur Apache 2.4.27 tournant sur la machine d'Ip `10.103.0.x`

Les sites préfixés par `www`, `get` et `put` sont accessibles par le port 80.  
Le site préfixé par `bank` est accessible par le port 443.

- Votre VM Linux sera connectée sur ce réseau.
- Les lignes suivantes seront rajoutées au fichier `/etc/hosts`:  
`10.103.0.x www.test.be`  
`10.103.0.x bank.test.be`  
`10.103.0.x get.test.be`  
`10.103.0.x post.test.be`
- Toutes les commandes seront lancées à partir cette VM.



# Ateliers de découverte

- **ATELIER 1:** Envoi d'une requête HTTP par telnet

```
#telnet www.test.be 80      → adresse et port d'écoute du serveur web
```

```
...  
GET / HTTP/1.1  
host: www.test.be
```

}

→ Requête du client

```
HTTP/1.1 200 OK  
Date: Wed, 13 Sep 2017 13:08:06 GMT  
Server: Apache/2.4.27 (Fedora) OpenSSL/1.1 ...  
...  
Connection: close  
Content-Type: text/html; charset=UTF-8
```

}

→ Réponse du serveur

```
<html>  
<head>  
<title> Site web de test </title>  
</head>  
<body>  
<h1> Site web de test. Ca marche !!! </h1>  
</body>  
</html>Connection closed by foreign host.  
#
```

}

→ Code HTML transmis au client et fermeture de la connexion effectuée par le serveur.



# Ateliers de découverte

- **ATELIER 1:** Envoi d'une requête HTTP par telnet (suite)

## Champs de la requête

GET / HTTP/1.1 → Méthode GET pour obtenir la ressource se trouvant à la racine du site.  
La requête est envoyée par le protocole HTTP 1.1

host: www.test.be

→ Le champ 'host' a été introduit dans la version 1.1 du protocole. Nécessaire pour que le client puisse toucher des sites hébergés par noms (voir Chapitre 'Serveur Web Apache')

## Champs de la réponse

HTTP/1.1 200 OK → Code de retour renvoyé par le serveur. Code 200 : requête accomplie avec succès.

Date: Wed, 13 Sep 2017 13:08:06 GMT → Date d'envoi de la réponse

Server: Apache/2.4.27 (Fedora) OpenSSL/1.1 ... → Informations sur le serveur  
...



# Ateliers de découverte

- **ATELIER 2:** Envoi de cette requête en HTTP 1.0

```
#telnet www.test.be 80
```

...

GET / HTTP/1.0 → La requête est envoyée via le protocole 1.0 (le champ 'host' serait ignoré même s'il était présent)

HTTP/1.1 200 OK → Le serveur répond via le protocole 1.1 en signalant que tout est accompli avec succès ...  
...

```
<html>
<head>
<title> Site web principal </title>
</head>
<body>
<h1> Site web principal. </h1>
</body>
</html>Connection closed by foreign host.
#
```



→ ... excepté que la page renvoyée n'est pas celle attendue.

Pourquoi ?  
voir Chapitre 'Serveur Web Apache'



# Ateliers de découverte

## • ATELIER 3: Codification d'une requête HTTP dans un script

```
# cat test.sh
#!/bin/bash
echo "open www.test.be 80"
sleep 2
echo "GET / HTTP/1.1"
echo "host: www.test.be"
echo
echo
sleep 2
```



→ contenu du script

```
# ./test.sh | telnet → pour l'exécuter
```



# Ateliers de découverte

- **ATELIER 4:** Requête vers une ressource qui n'existe pas

```
#telnet www.test.be 80
```

```
...
```

```
GET /notexist/ HTTP/1.1
host: www.test.be
```



→ On tente de récupérer la page d'accueil d'un site se trouvant dans le dossier 'notexist' du site racine...

```
HTTP/1.1 404 Not Found
```

→ Code de retour : 404 (Not Found)

```
...
```

La ressource demandée n'a pas été trouvée.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

```
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /notexist/ was not found on this server.</p>
</body></html>
```

→ Code HTML d'une page d'erreur standard transmise au client.

```
Connection closed by foreign host.
```

```
#
```



# Ateliers de découverte

- **ATELIER 5:** Redirection d'une requête

```
#telnet www.test.be 80
```

...

```
GET /service/ HTTP/1.1
host: www.test.be
```



→ On tente de récupérer la page d'accueil d'un site se trouvant dans le dossier 'service' du site racine...

```
HTTP/1.1 301 Moved Permanently
```

...

```
Location: http://www.test.be/staff/
```



Code de retour : 301

→ La ressource demandée a été déplacée dans le dossier 'staff'



Pour retourner la valeur du champ 'Location', le serveur a besoin de s'autoréférencer.  
C'est là que la directive 'ServerName' interviendra (voir Chapitre 'Serveur Web Apache').

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://www.test.be/staff/">here</a>. </p>
</body></html>
```

→ Code HTML d'une page de redirection standard transmise au client.

Connection closed by foreign host.

#



# Ateliers de découverte

- **ATELIER 5:** Redirection d'une requête (suite)

```
#telnet www.test.be 80
```

```
...
```

```
GET /staff/ HTTP/1.1  
host: www.test.be
```



→ Alors, tentons de récupérer la page d'accueil dans ce dossier...

```
HTTP/1.1 200 OK → Ca marche !
```

```
...
```

```
<html>  
<head>  
<title> Site web de notes de service </title> → Code HTML de cette page  
</head> d'accueil.  
<body>  
<h1> Site web des notes de service. Ca marche !!! </h1>  
</body>  
</html>Connection closed by foreign host.  
#
```

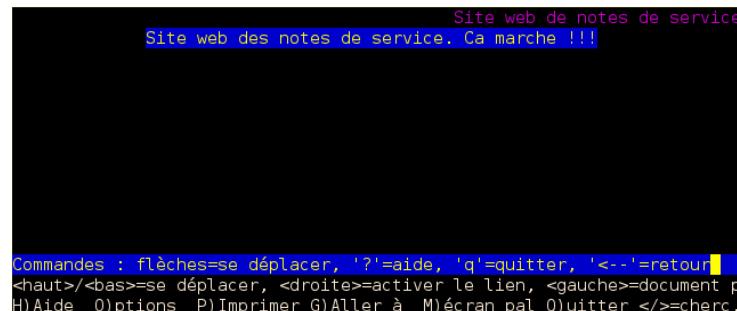


# Ateliers de découverte

- **ATELIER 5:** Redirection d'une requête (suite)

Tous les clients web sont programmés de façon à réitérer une requête lorsqu'ils reçoivent une réponse avec un champ 'Location'.

```
# lynx www.test.be/service
```



**URL saisie:** http://www.test.be/service



# Ateliers de découverte

- **ATELIER 6:** Accès à un site protégé

```
#telnet www.test.be 80
```

...

```
GET /controle/ HTTP/1.1
host: www.test.be
```

} → On tente de récupérer la page d'accueil d'un site se trouvant dans le dossier 'controle' du site racine...

**HTTP/1.1 401 Unauthorized** → Code de retour 401. On a besoin de s'identifier pour accéder à cette ressource.  
...

**WWW-Authenticate: Basic realm="Acces au site sous contrôle"**

... Le serveur renvoie dans sa réponse un champ 'WWW-Authenticate' indiquant au client le type d'authentification, ici: Basic (voir Chapitre 'Serveur Web Apache').

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.</p>
</body></html>
```

→ Code HTML de cette page d'erreur.

Connection closed by foreign host.

#



# Ateliers de découverte

- **ATELIER 7:** Refus d'accès à une ressource

```
#telnet www.test.be 80
```

```
...
```

```
GET /accueil.html HTTP/1.1 }  
host: www.test.be
```

→ On tente de récupérer la page accueil.html du site racine...

```
HTTP/1.1 403 Forbidden
```

```
...
```

→ Code de retour 403. L'accès à cette ressource est refusé.  
Le serveur Apache n'a probablement pas le droit de lire ce fichier (droit 'r' sous Linux).

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>403 Forbidden</title>  
</head><body>  
<h1>Forbidden</h1>  
<p>You don't have permission to access /accueil.html  
on this server.<br />  
</p>  
</body></html>  
Connection closed by foreign host.  
#
```

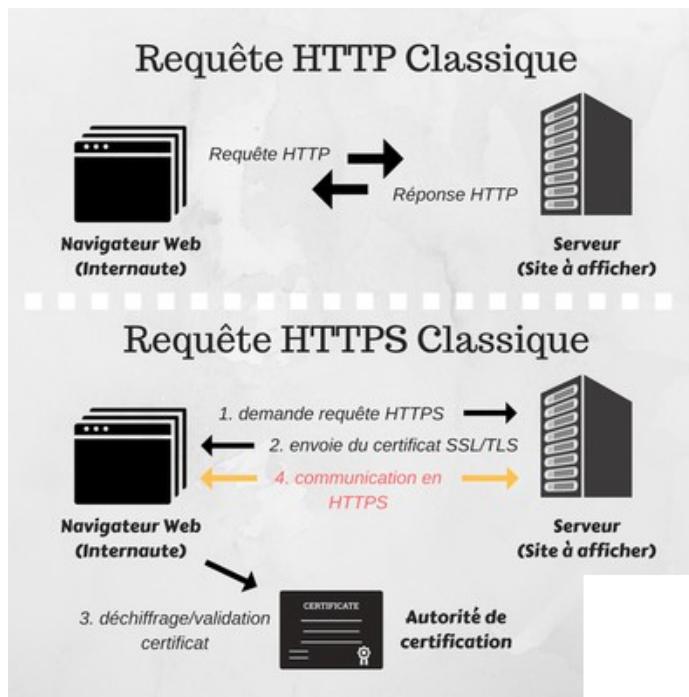
→ Code HTML de cette page d'erreur.



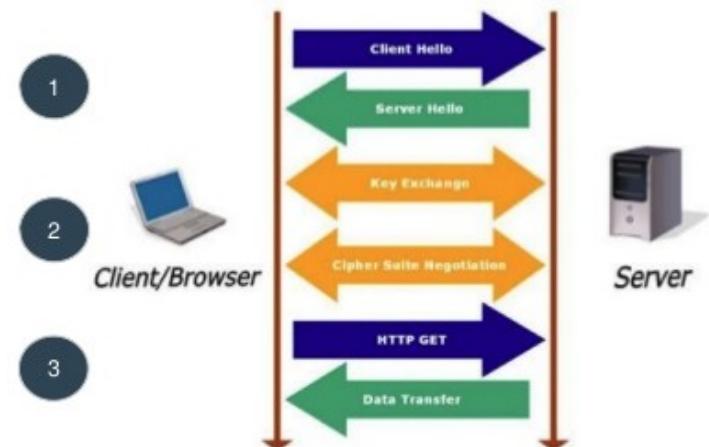
# Ateliers de découverte

- **ATELIER 8:** Accès sécurisé à un site

- Cas classique: l'accès à un site bancaire se fait via le protocole HTTPS.
- Les données échangées entre le client et le serveur sont chiffrées.



## FONCTIONNEMENT PROTOCOLE HTTPS



- 1. « Handshake »**
- 2. Echange des clés de chiffrement**
- 3. GET du client → Envoi de la page en mode crypté**

- Le serveur de test héberge un site sécurisé accessible via l'URL:

Le protocole HTTP (JL Gouwy) <https://bank.test.be> 248



# Ateliers de découverte

- **ATELIER 8:** Accès sécurisé à un site (suite)

```
#telnet bank.test.be 80
```

```
...
GET / HTTP/1.1
host: bank.test.be
```

HTTP/1.1 200 OK → Le serveur répond en signalant que tout est accompli avec succès ...

...

```
<html>
<head>
<title> Site web principal </title>           → ... excepté que la page reçue n'est pas celle
</head>                                              du site bancaire !!!
<body>
<h1> Site web principal. </h1>
</body>
</html>Connection closed by foreign host.
#
```



# Ateliers de découverte

- **ATELIER 8:** Accès sécurisé à un site (suite)

```
#telnet bank.test.be 443 → 443 étant le port d'écoute par défaut d'un serveur https.
```

```
...
GET / HTTP/1.1
host: bank.test.be
```

## HTTP/1.1 400 Bad Request

- ... Le serveur ne comprend pas la requête car il est configuré en https sur le port 443.
- ... Donc, toute requête entrant par ce port doit être initiée par un message 'hello' et non 'GET'.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
Reason: You're speaking plain HTTP to an SSL-enabled server port.<br />
Instead use the HTTPS scheme to access this URL, please.<br />
</p>
</body></html>
Connection closed by foreign host.
#
```

→ Code HTML de cette page d'erreur.



# Ateliers de découverte

- **ATELIER 8:** Accès sécurisé à un site (suite)

Dans ce cas, il est donc nécessaire d'utiliser un utilitaire de la bibliothèque 'openssl' se chargeant de réaliser tous les échanges nécessaires afin de pouvoir créer un canal sécurisé au travers lequel la conversation HTTP pourra circuler.

```
# openssl s_client -connect bank.test.be:443
```

...  
... } → Le certificat est téléchargé mais n'est pas validé. En effet, ici, nous travaillons avec un certificat auto-signé qui n'est pas reconnu par le client (voir Chapitre 'Serveur Web Apache').

```
GET / HTTP/1.1
```

**HTTP/1.1 400 Bad Request** → Le canal sécurisé n'est donc pas créé et l'échange HTTP ne peut donc se poursuivre.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>
closed
#
```

→ Code HTML de la page d'erreur.



# Ateliers de découverte

- **ATELIER 8:** Accès sécurisé à un site (suite)

Codification d'une requête HTTPS dans un script

```
# cat testhttps.sh
#!/bin/bash
echo "s_client -connect www.kernel.org:443"
sleep 2
echo "GET / HTTP/1.1"
echo "host: bank.test.be"
echo
echo
sleep 2
```

```
# ./testhttps.sh | openssl > out      → Pour l'exécuter
```



# Ateliers de découverte

- **ATELIER 8:** Accès sécurisé à un site (suite)

```
# ./testhttps.sh | openssl > out
```

```
...
```

```
# cat out
```

```
...
```

```
...
```

```
...
```

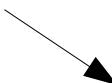
→ Le certificat est téléchargé et validé. En effet, ici, le certificat a été certifié conforme  
Par une autorité de certification reconnue par le client (voir Chapitre 'Serveur Web Apache').

```
HTTP/1.1 200 OK
```

```
Server: nginx
```

```
...
```

```
--- Code html de la page d'accueil du site ---
```



Ce code est chiffré par le serveur, puis transporté et déchiffré par le client.



# Ateliers de découverte

- **ATELIER 8:** Accès sécurisé à un site (suite)

curl (client URL library) permet de récupérer le contenu d'une ressource accessible par un réseau informatique

```
# curl https://bank.test.be
```

```
# curl -k -v https://bank.test.be
```



Pour' bypasser' la validation du certificat.

Mode bavard.

```
# curl -v http://www.test.be
```

...



# Ateliers de découverte

- **ATELIER 9:** Les connexions persistantes

Une connexion persistante garde la connexion réseau ouverte afin de faire passer plusieurs transactions entre le client et le serveur à l'intérieur de celle-ci.

Sans connexion persistante, les  $x$  objets (images, css...) d'une page seront transférés par  $x$  connexions réseaux différentes → lenteur



# Ateliers de découverte

- **ATELIER 9:** Les connexions persistantes (cp)

		<b>SERVEUR</b>	
		<b>Ouvre une CP (1)</b>	<b>N'ouvre pas de CP (2)</b>
<b>C L I E N T</b>	<b>Ne demande pas de CP</b>	Le serveur envoie la page et fermera la connexion après 30 secondes. Durant ce temps, le processus fils httpd est mobilisé en état d'attente au lieu de pouvoir servir des requêtes. On conseille une persistance allant de 2 à 5 sec. maximum.	Le serveur envoie la page et signale au client qu'il fermera immédiatement la connexion après l'envoi de la page.  <u>Réponse</u> <b>Connection: close</b>  Le serveur ferme la connexion directement après téléchargement de la page.
	<b>Demande une CP</b>  <u>Requête</u> <b>Connection: keep-alive</b>	Le serveur accepte la demande mais signale au client qu'il fermera la connexion après 30 sec.  <u>Réponse</u> <b>Keep-Alive : timeout=30, max=100</b> <b>Connection: Keep-Alive</b>  La connexion reste ouverte durant 30 sec. Puis est fermée par le serveur.	Le serveur n'accepte pas la demande du client et lui signale qu'il fermera immédiatement la connexion après l'envoi de la page.  <u>Réponse</u> <b>Connection: close</b>  Le serveur ferme la connexion directement après téléchargement de la page.

**(1) KeepAlive On**

KeepAliveTimeOut 30

MaxKeepAliveRequest 100

**(2) KeepAlive Off**

<https://httpd.apache.org/docs/2.4/fr/mod/core.html#keepalive>



# Ateliers de découverte

- **ATELIER 9:** Les connexions persistantes (suite)

Les clients HTTP 1.1 (ils le sont tous actuellement) envoient toujours leur requête en demandant une connexion persistante.

The screenshot shows a browser's developer tools Network tab with the 'Tout' filter selected. The table lists numerous requests made to the URL `http://www.apache.org/`. The 'En-têtes' (Headers) section is expanded, showing detailed information for each request. Two specific headers are highlighted with black ovals: `Accept-Encoding` and `Connection`. An arrow points from the `Connection` header in the response headers back to the `Upgrade-Insecure-Requests` header in the request headers, illustrating how the client asks for a persistent connection.

État	Méthode	URL de la requête
200	GET	/
200	GET	css?family=SF
200	GET	min.bootstrap
200	GET	styles.css
200	GET	ASF logo.p
200	GET	Support Ap
200	GET	acna-pres
200	GET	annualrep
200	GET	manifoldcl
200	GET	helix.jpg
200	GET	apex.jpg
200	GET	jquery-2.1.1.r
200	GET	bootstrap.js
200	GET	OAKsP294HTC
200	GET	glyphicon-ha
200	GET	cse.js?cx=00
200	GET	toadOcfmlt9b
200	GET	QQt14e8dy39
200	GET	jsapi?autoloa
200	GET	default+en.cs
200	GET	default+en.lj
204	GET	generate_204
200	GET	default.css
200	GET	async-ads.js
200	GET	Googlelog
200	GET	search_ba

URL de la requête : `http://www.apache.org/`  
Méthode de la requête : GET  
Adresse distante : 88.198.26.2:80  
Code d'état : 200 OK  
Version : HTTP/1.1

Filtrer les en-têtes

En-têtes de la réponse (0,399 Ko)

**Accept-Ranges** : "bytes"  
**Cache-Control** : "max-age=3600"  
**Connection** : "Keep-Alive"  
**Content-Encoding** : "gzip"  
**Content-Length** : "13235"  
**Content-Type** : "text/html"  
**Date** : "Wed, 13 Sep 2017 18:55:27 GMT"  
**Etag** : "'e6af-5591613961afa-gzip'"  
**Expires** : "Wed, 13 Sep 2017 19:55:27 GMT"  
**Keep-Alive** : "timeout=30, max=100"  
**Last-Modified** : "Wed, 13 Sep 2017 18:10:36 GMT"  
**Server** : "Apache/2.4.7 (Ubuntu)"  
**Vary** : "Accept-Encoding"

En-têtes de la requête (0,314 Ko)

**Host** : "www.apache.org"  
**User-Agent** : "Mozilla/5.0 (X11; Linux x86\_64; rv:52.0"  
**Accept** : "text/html,application/xhtml+xml,application/  
**Accept-Language** : "en-US,en;q=0.5"  
**Accept-Encoding** : "gzip, deflate"  
**DNT** : "1"  
**Connection** : "keep-alive"  
**Upgrade-Insecure-Requests** : "1"



# Autres champs

**Plus d'info:**

<http://abcdrfc.free.fr/rfc-vf/rfc2616.html>  
<https://www.w3.org/Protocols/rfc2616/rfc2616.html>

The screenshot shows a browser's developer tools Network tab with a list of requests and their responses. A specific response for 'helix.jpg' is highlighted, and its headers are expanded. Callouts from various parts of the headers point to explanatory text.

Etat	Méthode	URL
200	GET	/
200	GET	css?family=
200	GET	min.bootstrap
200	GET	styles.css
200	GET	ASF logo,p
200	GET	SupportA
200	GET	acna-pres
200	GET	annualrep
200	GET	manifoldct
200	GET	HELIx helix.jpg
200	GET	apex.jpg
200	GET	jquery-2.1.1.r
200	GET	bootstrap.js
200	GET	0AKsP294HTC
200	GET	glyphicon-ha
200	GET	cse.js?cx=00
200	GET	toadOcfmnl9b
200	GET	QQt14e8dy3s
200	GET	jsapi?autoloa
200	GET	default+en.cs
200	GET	default+en.li.j
204	GET	generate_204
200	GET	default.css
200	GET	async-ads.js
200	GET	googlelog.g
200	GET	search_ba

**En-têtes**      **Cookies**

**URL de la requête :** http://www.apache.org/  
**Méthode de la requête :** GET  
**Adresse distante :** 88.198.26.2:80  
**Code d'état :** 200 OK  
**Version :** HTTP/1.1

**En-têtes de la réponse (0,399 Ko)**

- Accept-Ranges :** bytes
- Cache-Control :** max-age=3600
- Connection :** Keep-Alive
- Content-Encoding :** gzip
- Content-Length :** 13235
- Content-Type :** text/html
- Date :** Wed, 13 Sep 2017 18:55:27 GMT
- Etag :** e6af-5591613961afa-gzip
- Expires :** Wed, 13 Sep 2017 19:55:27 GMT
- Keep-Alive :** timeout=30, max=100
- Last-Modified :** Wed, 13 Sep 2017 18:10:36 GMT
- Server :** Apache/2.4.7 (Ubuntu)
- Varv :** Accept-Encoding

**En-têtes de la requête (0,314 Ko)**

- Host :** www.apache.org
- User-Agent :** Mozilla/5.0 (X11; Linux x86\_64; rv:52.0)
- Accept :** text/html,application/xhtml+xml,application
- Accept-Language :** en-US,en;q=0.5
- Accept-Encoding :** gzip, deflate
- DNT :** 1
- Connection :** keep-alive
- Upgrade-Insecure-Requests :** 1

**Le client mettra en cache cette ressource durant 3600 sec.**

**L'entité transférée est compressée en format gzip. Le client possède l'algorithme de décompression (voir champ 'Accept-Encoding')**

**Taille du corps du message**

**Type et sous-type du média utilisé par le corps du message. Souvent en corrélation avec l'Accept de la requête.**

**Le client se présente.**

**Type de média préféré par le client**

**Si le serveur le possède, le document sera envoyé dans cette langue.**

**Le serveur pourra compresser des entités dans un algo. que le client comprend.**

**Conseil sur la sécurité:** [https://httpd.apache.org/docs/2.4/fr/misc/security\\_tips.html](https://httpd.apache.org/docs/2.4/fr/misc/security_tips.html)



# Les méthodes des clients

## • LES METHODES CLASSIQUES

GET	Obtenir la ressource située à l'URL spécifiée. Elle peut-être le contenu d'un fichier statique ou elle peut invoqué un programme qui génère des données (script CGI, PHP...).
HEAD	Obtenir la ressource située à l'URL spécifiée (la réponse ne contient que l'entête, et pas le contenu de la ressource).
POST	Envoi de données au programme situé à l'URL spécifiée (le corps de la requête peut être utilisé).
PUT	Ajouter une ressource sur le serveur.
DELETE	Suppression de la ressource située à l'URL spécifiée.

*Dans la pratique peu de serveurs autorisent les actions de type PUT et DELETE pour des raisons évidentes de sécurité.*



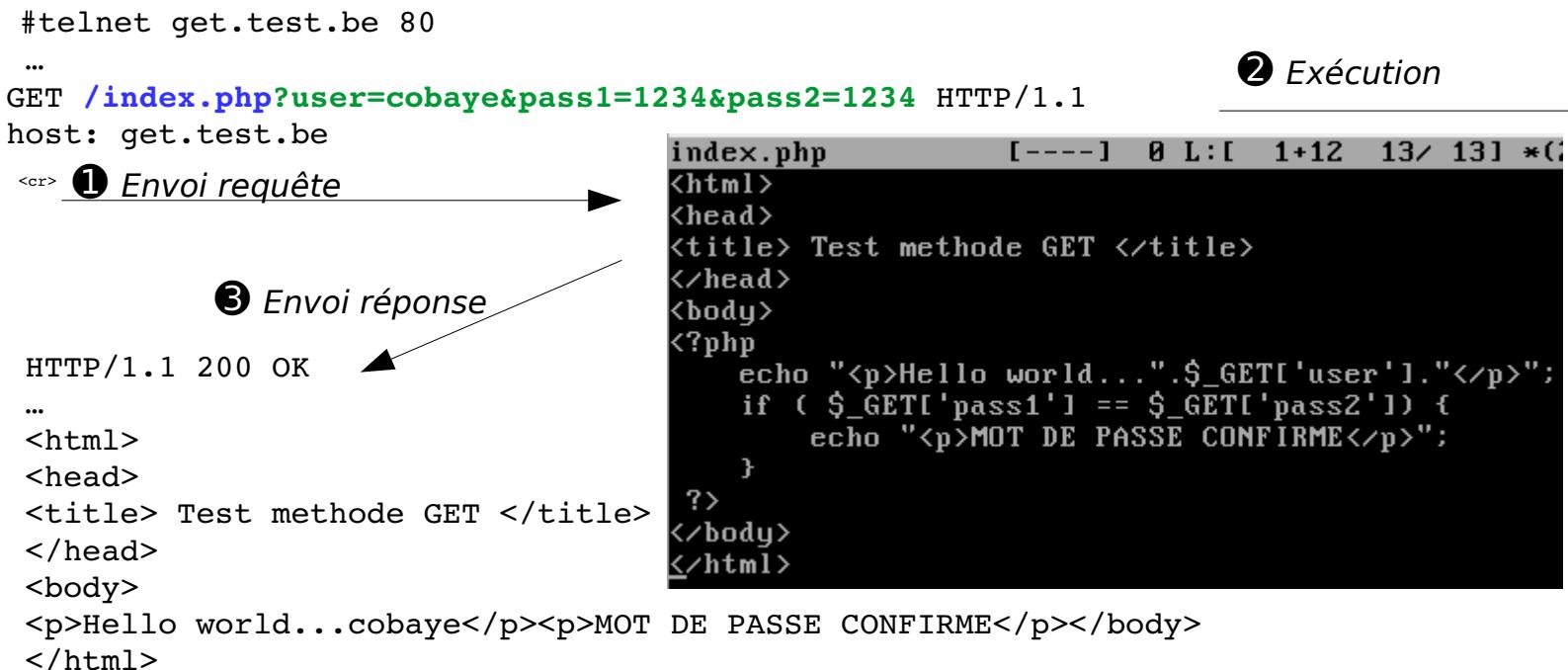
# Les méthodes des clients

- **LA METHODE GET**

- Pour récupérer une ressource statique ou dynamique sur le serveur.

Statique : ex. une page HTML

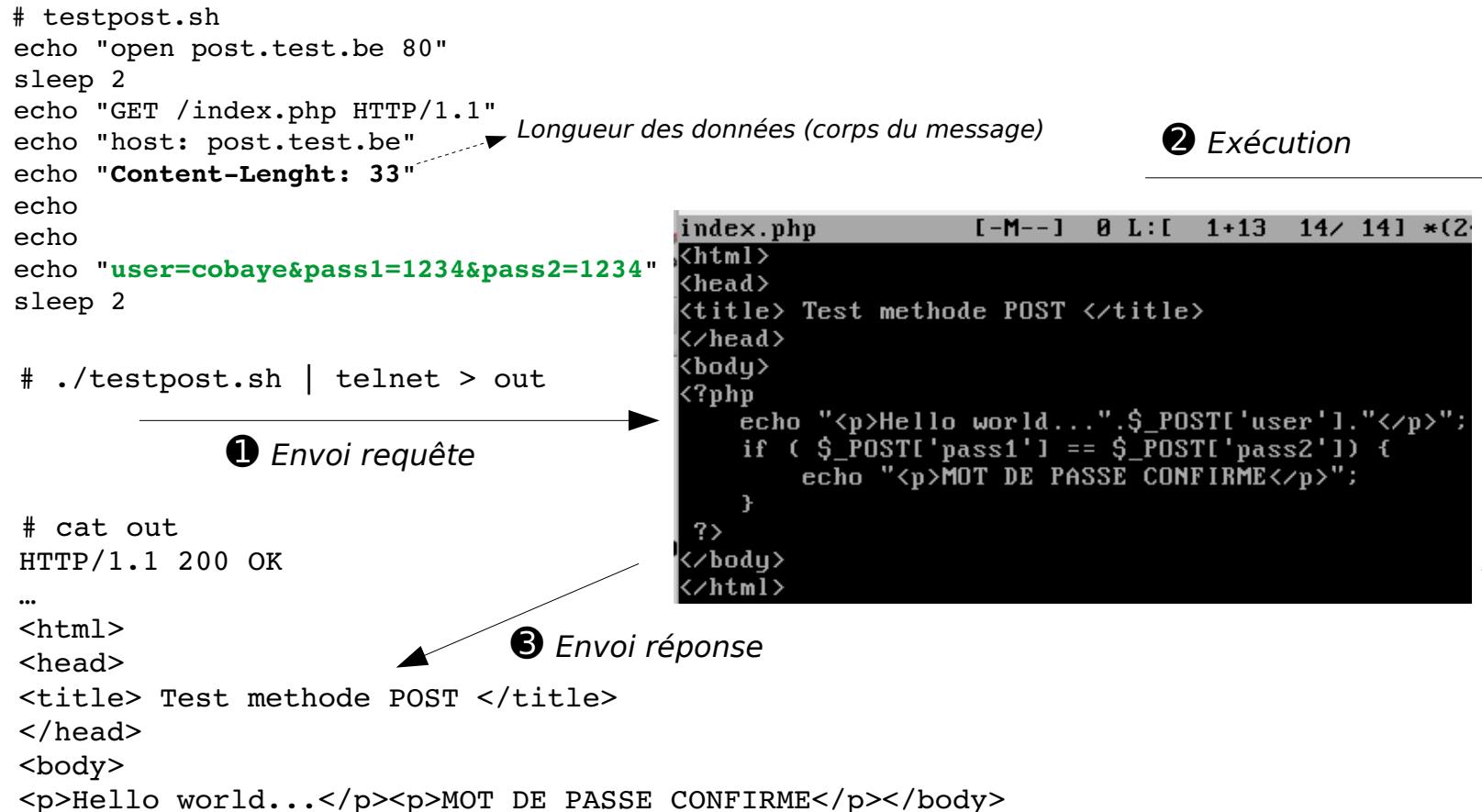
Dynamique : ex. une page HTML générée par un script PHP auquel on peut passer des arguments



# Les méthodes des clients

- **LA METHODE POST**

- Pour envoyer **des données** à des programmes sur le serveur.



# Les méthodes des clients

- **GET vs POST**

	Taille des données	Type de données	Invocation	Localisation des données	Visibilité dans l'URL du browser	Sniffing des données sensibles
GET	1024 car. maximum	ASCII	Souvent à partir d'un code HTML <sup>(1)</sup>	En argument dans la méthode de l'en-tête de la requête	Oui → danger si on passe des données sensibles (ex.mdp) <sup>(3)</sup>	Oui car le protocole HTTP circule en clair <sup>(3)</sup>
POST	Aucune restriction	ASCII ou binaire (image, son ...)	Souvent à partir d'un code HTML <sup>(2)</sup>	Dans le corps de l'en-tête de la requête	Non	Oui car le protocole HTTP circule en clair <sup>(3)</sup>

<sup>(1)</sup> `form method="get" action="/index.php"`

<sup>(2)</sup> `form method="post" action="/index.php"`

<sup>(3)</sup> Pistes pour sécuriser un échange GET ou POST de données sensibles :

- Salage (chiffrement) de ces données
- Utiliser HTTPS
- ...



# Ateliers d'espionnage

## • LA METHODE GET

Récupération via tshark les mots de passe passés en arguments.

### Terminal 2

```
#tshark -V -i eth0 port http > sniff.get  
...
```

### Terminal 1

```
# telnet get.test.be 80  
...  
GET /index.php?user=cobaye&pass1=1234&pass2=1234 HTTP/1.1  
host: get.test.be  
<cr>  
...
```

### Terminal 2

```
# cat sniff.get| grep "GET"  
GET /index.php?user=cobaye&pass1=1234&pass2=1234 HTTP/1.1\r\n  
...  
#
```



# Ateliers d'espionnage

## • LA METHODE POST

Récupération via tshark les mots de passe passés dans le corps de l'en-tête.

### Terminal 2

```
# tshark -V -i eth0 port http > sniff.post
...
...
```

### Terminal 1

```
# ./testpost.sh | telnet
...
...
```

### Terminal 2

```
# ^C
# n=`grep -n "Full request URI" sniff.post| cut -d: -f1` ; tail -lines=+$n
                                         sniff.post | more
[Full request URI: http://post.test.be/index.php]
[HTTP request 1/1]
Hypertext Transfer Protocol
\r\n
Data (35 bytes)

0000  75 73 65 72 3d 63 6f 62 61 79 65 26 70 61 73 73  user=cobaye&pass
0010  31 3d 31 32 33 34 26 70 61 73 73 32 3d 31 32 33  1=1234&pass2=123
...
#
#
```



# Les codes de retour

10x	Codes d'informations.
20x	L'opération s'est correctement effectuée. Le plus courant est le code 200 (OK).
30x	L'objet demandé a été déplacé (ex. redirection).
40x	Lorsque le client a commis une erreur. Erreur '404 Not found' lorsque l'objet demandé n'existe plus...
50x	Erreur du côté du serveur tel qu'un bug dans un script CGI.

Plus d'info: [http://fr.wikipedia.org/wiki/Liste\\_des\\_codes\\_HTTP](http://fr.wikipedia.org/wiki/Liste_des_codes_HTTP)



# Références

- **HTTP précis & concis**

Editions O'Reilly, Paris 2000

Clinton Wong

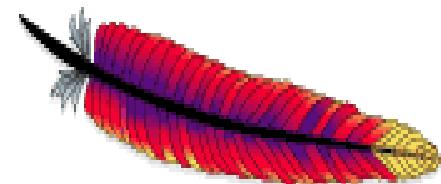
Traduction de Laurent Bourdron

ISBN 2-84177-115-6



## Laboratoire de sécurité internet

# Serveur Web Apache



**Jean-Louis Gouwy**



# Plan

- Introduction (Historique / Caractéristiques)
- Le modèle client-serveur (Les échanges)
- Fonctionnalités
- Architecture (Vue d'ensemble / Noyau / Modules / Filtres)
- Installation (Par les sources / Par les binaires / Architecture sous Fedora 26 )
- Configuration (Structure de l'httpd.conf / Structure de l'httpd.conf sous Fedora 26 / Le contexte des directives)
- Environnement principal (ServerName / ServerRoot / DocumentRoot / ServerAdmin / ServerTokens/ Listen / ErrorDocument)
- Exercice 1
- Contrôler Apache (L'arbre des processus / Création des instances de httpd  
Les directives: MinSpareServers - MaxSpareServers – StartServers – MaxRequestWorkers – ServerLimit - User - Group)
- Les sites perso
- Les redirections simples
- Les index de répertoires
- Exercice 2



# Plan

- Protection des sites web (Objectifs / Contrôle sur l'origine des clients / Le conteneur <Directory> / Contrôle par authentification / Autres types d'authentification)
- Exercice 3
- Hébergement virtuel (Introduction / Principe / Par Ip / Par nom )
- Exercice 4
- Les fichiers journaux (Le journal des erreurs / Le journal des accès / Les hôtes virtuels)
- Exercice 5
- Les certificats (Objectifs / SSL / Structure générale / Fonctionnement Question de confiance / Signature / Implémentation/ La redirection SSL)
- Exercice 6
- Références



# Introduction

## • HISTORIQUE

- 1965: Naissance en Suisse de l'idée originale de l'hypertexte (réseau d'un ensemble de documents informatiques liés entre eux) puis de l'htpd (http daemon).
- 1992: 26 serveurs Web.
- 1995: Démarrage du projet Apache (version 1.0).
- 1996: Apache devient le serveur Web le plus répandu.
- 1999: Constitution de l'ASF (Apache Software Foundation).  
*<http://www.apache.org> (ASF)*  
*<http://httpd.apache.org> (serveur Apache)*

Le serveur Apache tient son nom d'une des plus fières tribus indiennes dont la vigueur et la faculté d'adaptation n'était plus à prouver.



# Introduction

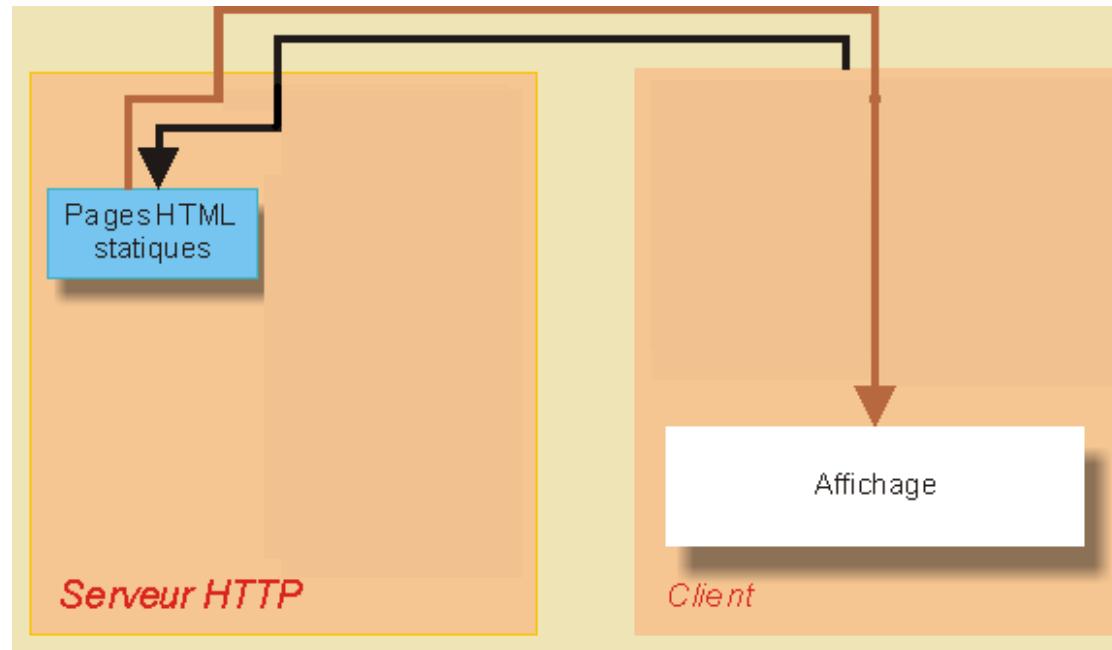
## • CARACTERISTIQUES

- Open Source et gratuité.
- Multi-plateformes (Unix- Unix Like - Linux - Windows – MacOs).
- C'est le serveur web 'opensource' le plus répandu.  
Source: <http://www.netcraft.com>
- Modulable, fiable, performant, sécurisé et extensible.
- Supporte les protocoles: HTTP / HTTPS (le plus souvent), POP3, FTP ...



# Le modèle client-serveur

- **LES ECHANGES: Demande d'une page html 'statique'**

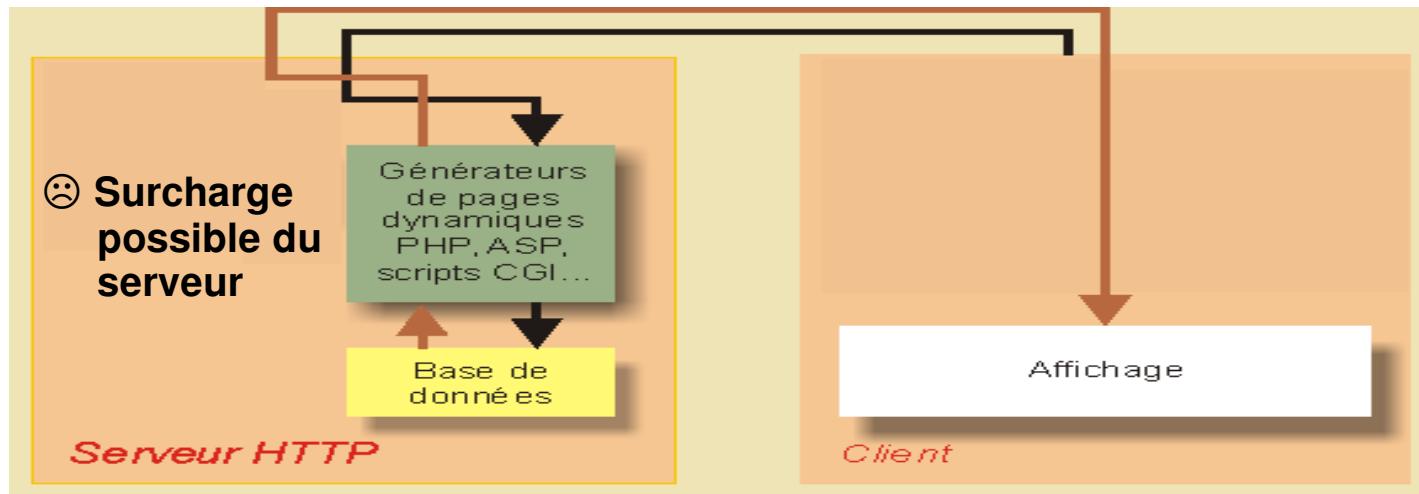


Ici, tout le contenu est défini dans la page HTML demandée.  
Le serveur l'envoie tel quel au client qui n'a plus qu'à l'interpréter et l'afficher.



# Le modèle client-serveur

- **LES ECHANGES: Demande d'une page html 'dynamique' (server side)**



- Les scripts CGI construisent totalement un flux HTML au moment de leur appel. On utilise de plus en plus souvent le C (compilé) ou le PERL (interprété).  
ou
- PHP (libre) ou ASP (Microsoft) qui sont des langages spécialisés et puissants.

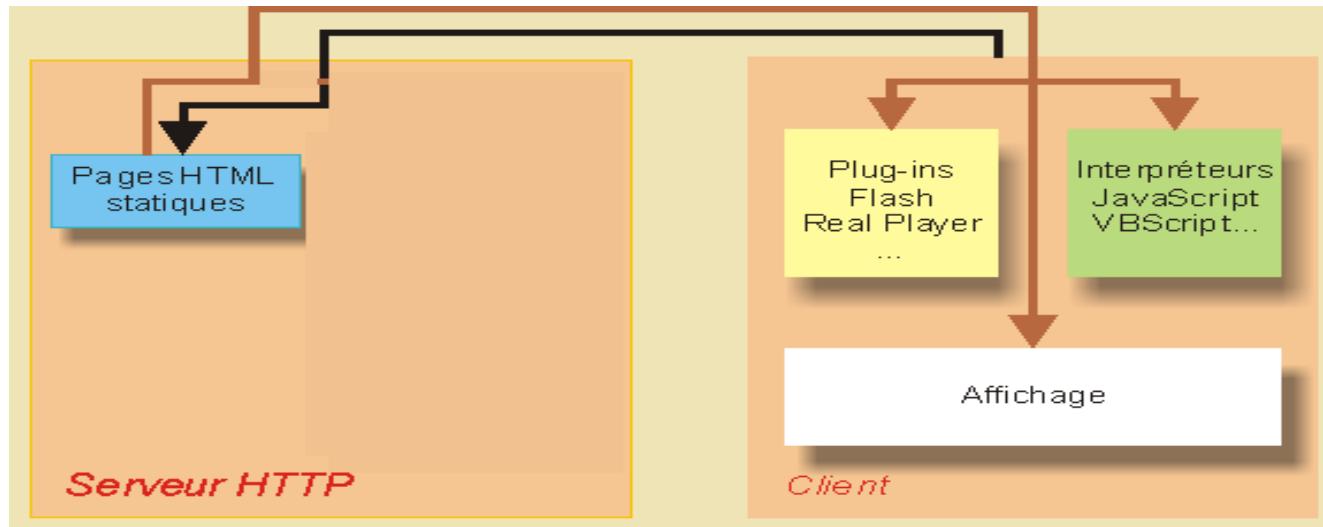
### Exemples:

- Demande d'un résultat calculé par le serveur et dont les données sont fournies par le client.
- Demande de données qui se trouvent dans une base de données hébergée côté serveur.



# Le modèle client-serveur

- **LES ECHANGES: Demande d'une page html 'dynamique' (client side)**



- ☺ Pas de surcharge sur le serveur.  
Pages animées.
- ☹ Incompatibilité des navigateurs.  
Failles de sécurité possibles côté client.

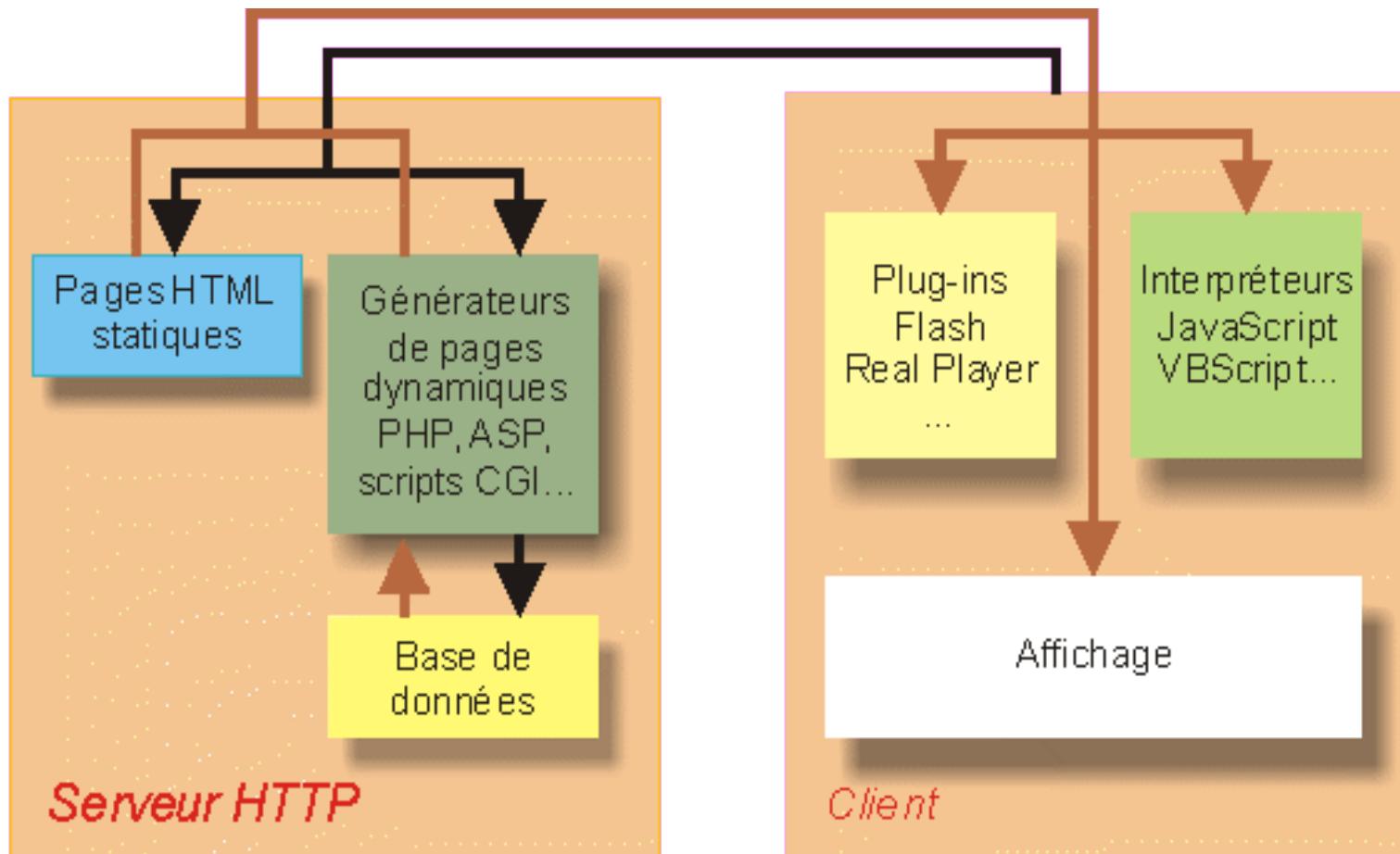
### Exemples:

- Contrôler la validité de données avant de les envoyer au serveur.
- Effectuer un traitement local pour afficher un résultat.
- Animations quelconques.



# Le modèle client-serveur

- **LES ECHANGES: Modèle complet**



# Fonctionnalités

- **Conformité aux standards:**  
Entièrement conforme aux standards HTTP/1.1 – RFC 2616).
- **Scalabilité:**  
Permet d'héberger un grand nombre de sites web sur une même machine sans voir les performances diminuer de manière cruciale.
- **Objets dynamiques partagés (shared objects):**  
Modules pouvant être compilés séparément du noyau et activés au démarrage d'Apache.
- **Personnalisation:**  
Programmation (en C ou en Perl) de modules personnels via l'API d'Apache.
- **Programmation:**  
Permet la programmation côté serveur (PHP, Perl, servlets Java, Java Server Page, Active Server Pages, CGI, FastCGI, Server-Side Includes)



# Fonctionnalités

- **Serveur mandataire:**

Apache peut être configuré en proxy serveur à l'aide de son module mod\_proxy.

- **Sécurité:**

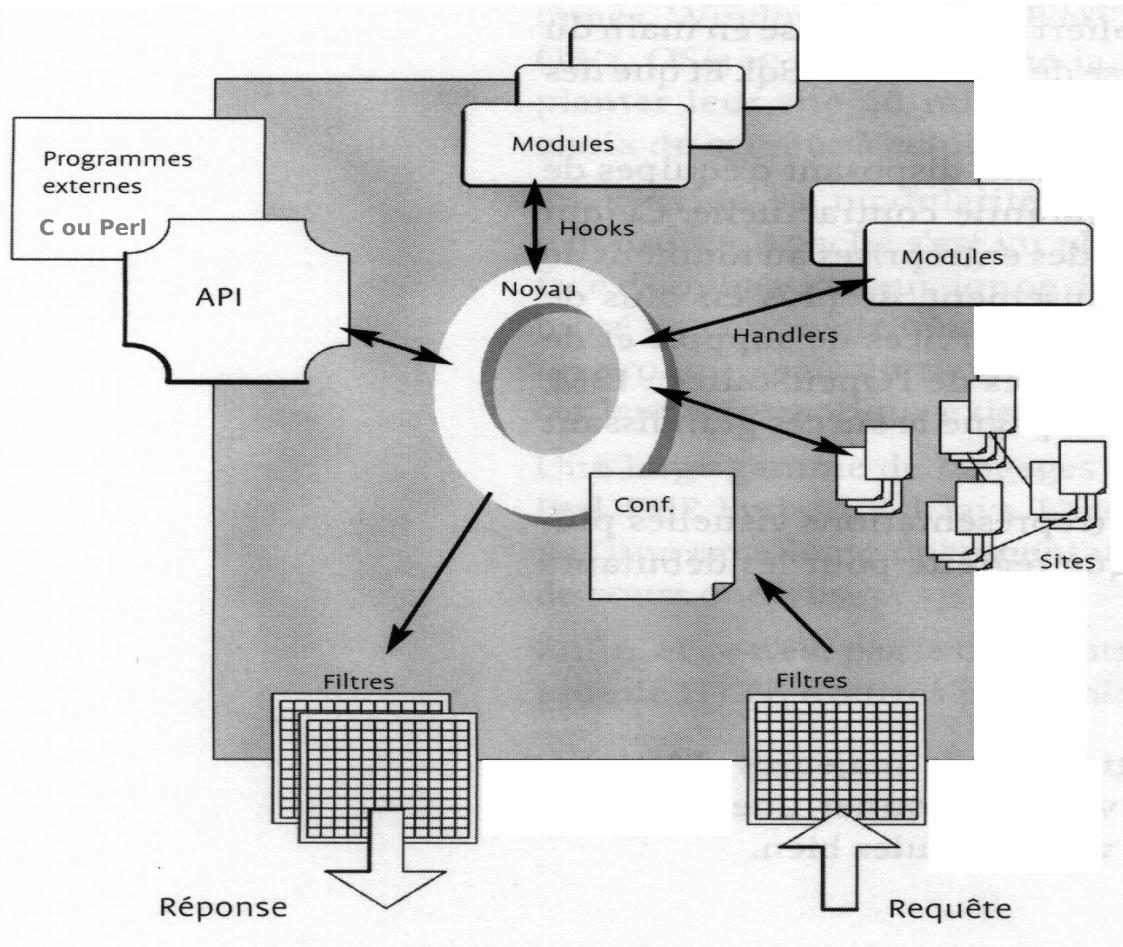
Par différentes méthodes d'authentification de l'utilisateur (fichier plats ou bases de données dbm-mysql-...)

Par l'autorisation du chiffrement sur Internet via l'échange de certificats numériques (SSL)



# Architecture

- VUE D'ENSEMBLE



# Architecture

- **Le noyau**

- Contient toutes les fonctionnalités de base du serveur (ex. directive Listen).
- Il distribue le travail aux modules ou aux programmes externes.
- Les documents sont extraits du ou des sites dont il a la charge.
- Il reçoit et comprend une requête.
- Il prépare et envoie une réponse.
- Il connaît ses ressources disponibles (modules disponibles, liste et localisation des sites à gérer ...) grâce à son fichier de configuration (httpd.conf).
- Pour chaque module qu'il va solliciter, il dispose d'un jeu de directives.



# Architecture

- **Les modules**

- Ils étendent les fonctionnalités du noyau.
- Ils peuvent être :
  - . Standards: Maintenus par l'ASF (ex. `mod_auth_basic`)  
Font partie de la distribution d'Apache  
(<http://httpd.apache.org/docs/2.4/fr/mod>)
  - . Tiers: Maintenus par des tiers (ex. `mod_auth_oracle`)  
Disponibles sur la toile.
- Ils peuvent être installés:
  - . Lors de la compilation (ou de l'installation) de la distribution d'Apache (modules standards)
  - . Lors d'une compilation séparée (modules tiers) via l'utilitaire `apxs`  
`# path_to/apxs -cia module.c` (-cia : compile, installe et active)



# Architecture

- **Les modules (suite)**

- Le choix des modules à utiliser se fait dans le fichier de configuration principal. Deux directives sont disponibles pour gérer les modules :

LoadModule : qui permet de charger un module au démarrage

<IfModule> : qui permet d'activer certaines parties du fichier de configuration si un module a été chargé.

- httpd -l Pour voir la liste des modules compilés dans le cœur d'Apache. Ce n'est pas la liste des modules chargés dynamiquement via la directive LoadModule.

Compiled in modules:

core.c

http\_core.c

mod\_so.c



*Doivent toujours être présents.*

→ *Permet le bon fonctionnement de la directive LoadModule (chargement de modules de type Shared Object)*

httpd -L | grep -i userdir

Pour connaître le module correspondant à chaque directive (ici la directive 'UserDir').



# Architecture

- **Les filtres**

Les filtres sont des modules standards ou tiers que vous choisissez d'incorporer ou non à Apache.

- . soit à l'entrée, sur le chemin des demandes entrantes
- . soit en sortie, sur le chemin des réponses sortantes

Par exemple un module de compression des données qui:

- . en entrée décomprime des requêtes compressées par un browser
- . en sortie compresse l'envoi de tous les documents du site



# Installation

- **Par les sources**

- Hors cadre du cours.
- Plus d'info: <http://httpd.apache.org/docs/2.4/fr/install.html>



# Installation

- **Par les binaires**

- Via la commande d'installation automatique de la distribution.
- Ce sera la retenue pour la suite du cours malgré que cette installation est moins souple que par compilation des sources (ciblage des modules plus difficile)...  
... par contre, elle offre un outil de désinstallation.

CentOS 6 : `yum install httpd -y` (Apache version 2.2)

CentOS 7 : `yum install httpd -y` (Apache version 2.4.6)

Fedora 26 Server avec Apache embarqué.

(Apache version 2.4.25) :

Nous retiendrons cette distribution car :

- la version d'Apache offerte est de la dernière branche 2.x
- de plus, le protocole HTTP 2 est correctement supportée à partir d'Apache 2.4.23.



# Installation

- **Architecture sous Fedora 26**

**/etc/httpd/** → dossier contenant l'ensemble des fichiers de configuration.

**conf/httpd.conf** → fichier principal de configuration.

**conf.d** → dossier contenant les fichiers secondaires de configuration.

**conf.modules.d** → dossier contenant les fichiers de lancement des modules.

**/var/www/** → dossier contenant les données du site par défaut.

**cgi-bin**: dossier (vide) contenant les scripts.

**html**: dossier (vide) contenant les pages du site par défaut.

**/var/log/httpd/** → dossier contenant les journaux

**access\_log** → journal des accès aux pages traitées par le serveur.

**error\_log** → journal des erreurs.



# Installation

- **Architecture sous Fedora 26**

**/user/share/**

- error → dossier contenant les pages affichées en cas d'erreur
- icons → dossier contenant quelques icônes
- manual → dossier contenant la documentation

**/usr/lib64/httpd/modules** → dossier contenant le binaire des modules (.so)

**/usr/sbin/httpd** → le daemon Apache

Remarques:

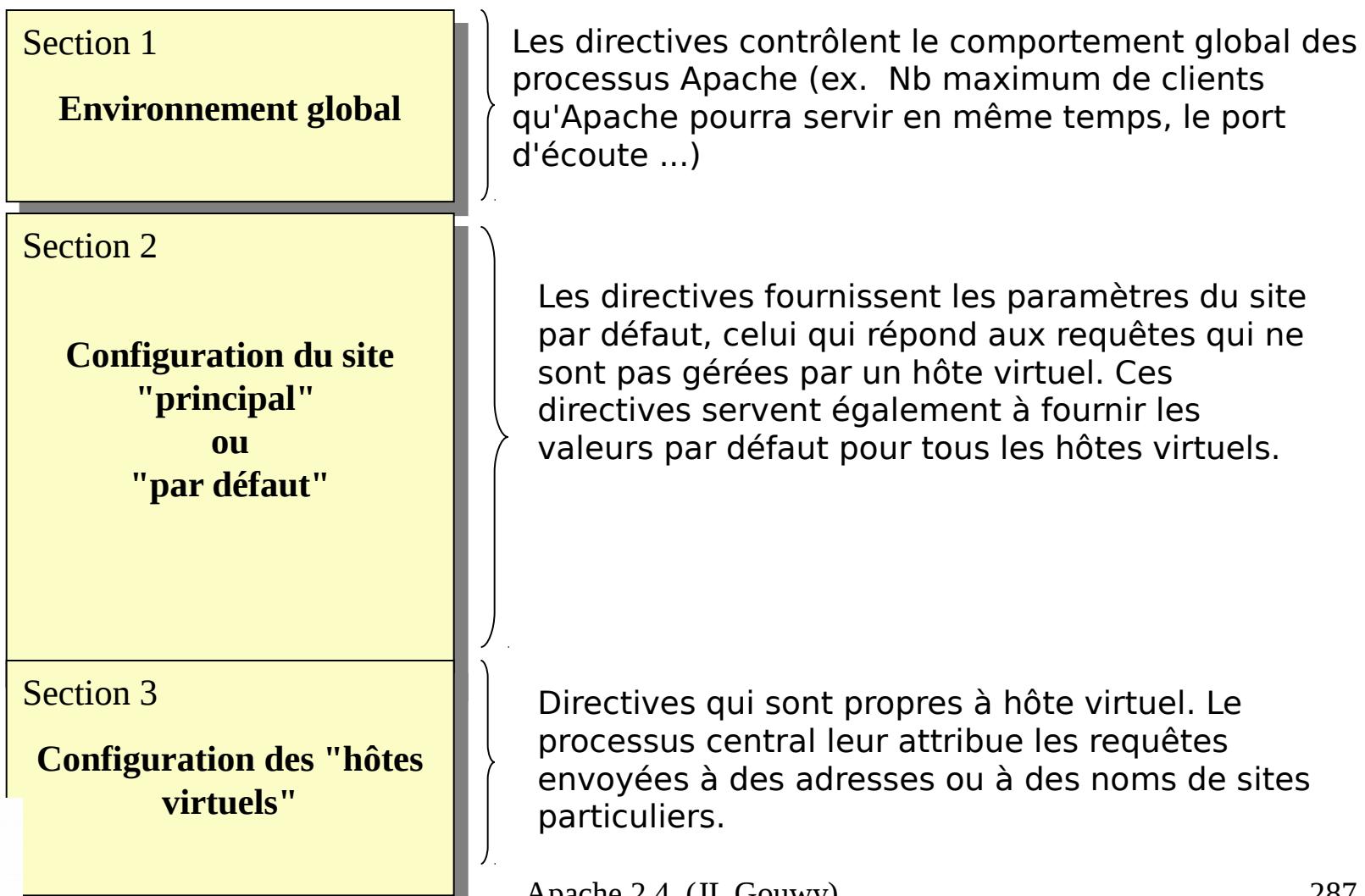
```
# systemctl start httpd.service  
→ lancement d'Apache (port d'écoute par défaut: 80)
```

```
# systemctl enable httpd.service  
→ sera lancé au démarrage du système
```



# Configuration

- **Structure de l'apache.conf**



# Configuration

- **Structure de l'httpd.conf sous Fedora 26**

```
...  
...  
Include conf.modules.d/* .conf ●  
...  
# 'Main' server configuration  
...  
IncludeOptional conf.d/* .conf ●
```

Chargés par ordre alphabétique

Il est préférable de ne pas le modifier.

On préfèrera la création d'un fichier spécifique à nos besoins dans le dossier /etc/httpd/conf.d.

Cela permet de ne pas bloquer les mises à jour éventuelles du fichier principal et simplifiera grandement les migrations (il suffit de récupérer notre fichier de configuration).



# Configuration

- **Le contexte des directives**

L'aide officielle d'Apache classe les directives dans 4 catégories

<http://httpd.apache.org/docs/2.4/fr/mod/quickreference.html>

<b>s</b>	server config
<b>v</b>	virtual host
<b>d</b>	directory
<b>h</b>	.htaccess



# Configuration

- **Le contexte des directives (suite)**

- Contexte général du serveur (server config - s)

- Agit sur tout le serveur.

ex.	StartServers	(uniquement dans ce contexte)
	ServerName	(dans ce contexte mais peut être redéfinie pour chaque hôte virtuel)

- Contexte hôte virtuel (virtualhost - v)

- De nombreuses directives d'hôte virtuel (définies dans le conteneur <VirtualHost>) surchargent celles qui sont générales au serveur.

ex.	ServerName
	DocumentRoot



# Configuration

- **Le contexte des directives (suite)**

- Contexte conteneur (directory - d)

Inclut les directives qui ne peuvent s'appliquer que dans un des 3 conteneurs (<Directory>, <Files> et <Location>) et dont la portée est limitée à ce conteneur.

ex.      Require

- Contexte .htaccess (.htaccess - h)

Sont traitées comme celles d'un conteneur <Directory> de l'httpd.conf. La principale différence est que les directives d'un fichier htaccess peuvent être désactivées au moyen de la directive AllowOverrides dans l'httpd.conf.



# L'environnement principal

- **La directive ServerName**

Apache doit toujours pouvoir déterminer le nom d'hôte de la machine sur laquelle il tourne car il l'utilise pour créer des URL d'autoréférence.

Exemple: ServerName www.mysite.be

- **La directive ServerRoot**

Répertoire dans lequel les fichiers de configuration, les logs et les modules sont gardés. Ce nom de dossier servira à préfixer tout chemin relatif rencontré dans l'htpd.conf.

Exemple: ServerRoot "/etc/htpd"



# L'environnement principal

- **La directive DocumentRoot**

Dossier dans lequel les documents du site par défaut sont déposés.  
C'est donc ce dossier qui contient les fichiers qu'Apache fournit lorsqu'il reçoit des requêtes avec l'URL /.

Exemple: DocumentRoot "/var/www/html"

- ☞ Le changement de nom du DocumentRoot doit aussi être effectué dans Son conteneur <Directory ...>, qui regroupe toutes les directives s'appliquant à DocumentRoot et ses sous-répertoires.

- **La directive ServerAdmin**

Adresse mail du webmaster qui pourrait s'afficher sur certains documents construits par le serveur et renvoyés au client en cas d'incidents.  
Valable si la directive **ServerSignature** est à l'état on permettant alors l'ajout de cette information en bas de page.

Exemple: ServerAdmin webmaster@mysite.be



# L'environnement principal

- **La directive ServerTokens**

Permet de contrôler le contenu de l'en-tête Server inclus dans la réponse envoyée au client.

## Exemples:

ServerTokens Prod[uctOnly]

➔ Le serveur renvoie (par ex.): Server: Apache

ServerTokens Major

➔ Le serveur renvoie (par ex.): Server: Apache/2

ServerTokens Minor

➔ Le serveur renvoie (par ex.): Server: Apache/2.0

ServerTokens Min[imal]

➔ Le serveur renvoie (par ex.): Server: Apache/2.0.41

ServerTokens OS

➔ Le serveur renvoie (par ex.): Server: Apache/2.0.41 (Unix)

ServerTokens Full (valeur par défaut)

➔ Le serveur renvoie (par ex.): Server: Apache/2.0.41 (Unix) PHP/4.2.2



# L'environnement principal

- **La directive Listen**

Pour définir les adresses IP et les numéros de ports sur lesquels Apache attend et reçoit les connexions des clients.

Exemples:

Listen 80 → Apache écoute sur toutes les interfaces sur le port 80.

Listen 10.0.0.7:80 → Apache écoute sur le port 80 sur l'interface d'IP 10.0.0.7.

Listen 80 → Apache écoute sur le port 80 et 8080 sur toutes les interfaces.

Listen 8080

Listen 192.168.1.1:80 → Apache répond aux requêtes http sur l'interface interne  
Listen 216.180.25.168:443 et aux requêtes https (connexions SSL) sur l'interface publique.



# L'environnement principal

- **La directive ErrorDocument**

Pour remplacer les pages d'erreur standards envoyées au client en cas de problème.

Exemples:

ErrorDocument 403 "Vous n'êtes pas autorisé à lire cette page !

*Ici on affiche simplement un texte adapté à l'erreur.*

ErrorDocument 401 /missing.html

*Ici on affiche une page html sensée se trouver à la racine du site web.*

ErrorDocument 500 http://www.bidon.com/erreur.html

*Ici on affiche une page html extérieure au site.*



ErrorDocument 401 nécessite toujours une URL interne.



# Exercice 1

- **Un serveur simple**

- Clonez le .vdi de la machine Fedora contenant une version supérieure à 2.4.23 du serveur Apache.
- Créez une MV 'apache' liée au disque cloné.
- Prefixez tous les fichiers d'extension .conf du dossier /etc/httpd/conf.d par '01-' et créez les fichier 01-main.conf et 00-server.conf
- Configurez le serveur pour:
  - . qu'il écoute sur le port 80 sur toutes les interfaces
  - . qu'il présente une page d'accueil index.html lors d'une requête vers l'URL de ce site (inventez son contenu)
  - . qu'il affiche une page html personnalisée en cas d'erreur 404
- Vérifiez la configuration du serveur.



# Exercice 1

- **Un serveur simple (suite)**

- f) Testez votre serveur pour vérifier si la page d'accueil est bien offerte:
  - . à l'aide de l'utilitaire telnet
  - . à l'aide d'un navigateur quelconque
- g) Reconfigurez Apache pour qu'il écoute cette fois sur le port 8080:
- h) Testez votre serveur:
  - . à l'aide de l'utilitaire telnet
  - . à l'aide d'un navigateur quelconque
- i) Compilez, installez et testez un module tiers (`mod_pony`)



# Contrôler Apache

- **L'arbre des processus**

Apache lance plusieurs daemons en parallèle; ceux-ci se trouvent en permanence à l'écoute du réseau afin de pouvoir répondre rapidement à un grand nombre de requêtes simultanées.

```
# service httpd restart
# ps -ef
```

			...
root	<b>2116</b>	1	S
apache	2119	<b>2116</b>	S
apache	2120	<b>2116</b>	S
apache	2121	<b>2116</b>	S
apache	2122	<b>2116</b>	S
apache	2123	<b>2116</b>	S
apache	2124	<b>2116</b>	S
apache	2125	<b>2116</b>	S
apache	2126	<b>2116</b>	S

0:00	/usr/sbin/httpd

*Processus principal.  
C'est lui qui reçoit les requêtes  
et les distribue à ses fils.*

*Maximum  
256 instances (fils).*



# Contrôler Apache

- **Création des instances de httpd**

Plus d'info.: <http://httpd.apache.org/docs/2.4/fr/misc/perf-tuning.html>

Les instances sont créées en fonction du type de module Multi-Processus (MPM) chargé :

- **Le MPM prefork:** chaque processus enfant possède un seul thread et chaque processus gère une seule connexion à la fois.
  - ☺ Aussi rapide qu'en 'worker'.  
Stable et universel (utilisable avec les modules tiers qui ne supportent pas le threading et compatible avec des logiciels ou OS anciens).
  - ☹ Plus gourmand en mémoire qu'en 'worker'.
- **Le MPM worker:** chaque processus enfant possède plusieurs threads et chaque thread gère une seule connexion à la fois.
  - ☺ Moins gourmand en mémoire qu'en 'prefork'.
  - ☹ Pas universel
- **Le MPM event** utilise les threads, mais il a été conçu pour traiter davantage de requêtes simultanément.
  - ☺ Moins gourmand en mémoire qu'en 'prefork' et plus rapide qu'en 'worker'
  - ☹ Pas universel



# Contrôler Apache

- **Les directives MinSpareServers / MaxSpareServers / StartServers / MaxRequestWorkers / ServerLimit**

**MinSpareServers:** Nombre minimal d'instances de serveurs.

**MaxSpareServers:** Nombre maximum d'instances de serveurs.

**StartServers:** Nombre de serveurs supplémentaires créés au démarrage d'Apache.

**MaxRequestWorkers / ServerLimit:**

Limite le nombre de processus qui peuvent tourner simultanément (chaque connexion cliente en utilise un).

Exemple: Voir exercice 2



# Contrôler Apache

- **Les directives User et Group**

Utilisateur et groupe Linux sous lequel s'exécuteront les processus fils d'Apache chargés de répondre aux requêtes des clients.

Le processus maître doit être lancé sous le compte root pour pouvoir changer le user et le group de ses processus fils.

Exemples:

User apache

Group apache



# Les sites perso

- Il est possible de permettre aux utilisateurs disposant d'un compte sur le serveur de posséder leur propre site.
- Pratique très courante parmi les FAI qui proposent l'hébergement de pages web de leurs clients.
- Cette fonctionnalité est fournie par le module standard mod\_userdir.

```
LoadModule userdir_module ...
```

- Elle est activée par la directive Userdir.

```
<IfModule mod_userdir.c>
    #UserDir disabled
    UserDir public_html
</IfModule>
```

*Si le module 'userdir' est chargé, alors un utilisateur du système pourra y héberger son site Web dont la racine se trouvera dans le dossier 'public\_html' de sa home directory.*

- Le site sera accessible via l'URL: `http://ip_serveur/~login_utilisateur`  
Exemple: `http://www.mysite.be/~jean`



# Les sites perso

- Autres formes de la directive UserDir :

UserDir disabled <user1 user2 ...>

Désactive la gestion des sites perso pour la liste des utilisateurs indiquée.

UserDir disabled

Désactive la gestion de tous les sites perso.

Souvent utilisée avant une directive UserDir enabled.

UserDir enabled <user1 user2 ...>

Active la gestion des sites perso pour la liste des utilisateurs indiquée.

Exemple:

UserDir disabled

UserDir enabled jean louis



# Les redirections simples

- La directive standard Alias permet d'accéder facilement à des documents HTML en dehors de l'arborescence DocumentRoot.
- Cette fonctionnalité est fournie par le module standard mod\_alias.

```
LoadModule alias_module ...
```

- Soit accéder à la page HTML /usr/share/doc/HTML/index.html via le site www.mysite.be par redirection:

```
Alias /CentOS "/usr/share/doc/HTML/"
```

```
<Directory "/usr/share/doc/HTML">  
    Require all granted  
</ Directory>
```



*Car la cible est dans un dossier situé en dehors de l'arborescence du site web  
→ permettre explicitement l'accès à ce dossier.*

Cette page sera accessible via l'URL:

<http://www.mysite.be/CentOS>



# Les index de réertoires

- **Recherche d'une page d'accueil**

Apache est capable de retrouver la page `index.html` sans pour autant que celle-ci soit indiquée dans l'URL.

Exemple:

`http://www.mysite.be/cours`

équivaut à:

`http://www.mysite.be/cours/index.html`



# Les index de répertoires

- **Recherche d'une page d'accueil (suite)**

- C'est le module standard `mod_dir` qui recherche et fournit une page (`index.html` par défaut) susceptible de se trouver dans le répertoire indiqué dans l'URL.
- Si la directive `DirectoryIndex` est présente, elle permettra d'ajouter des références à d'autres pages.

```
DirectoryIndex index.html index.htm index.php
```



# Les index de répertoires

- **Recherche d'une page d'accueil (suite)**

- Si aucune page d'accueil n'est trouvée, le module mod\_autoindex (piloté par sa directive IndexOptions) crée un index des fichiers du répertoire concerné.

## Exemples:

IndexOptions None (indexation classique)

IndexOptions FancyIndexing (indexation au look plus agréable)

IndexOptions FancyIndexing VersionSort

(idem + tri des entrées contenant des numéros de versions)

- Pour supprimer l'indexation sur un dossier particulier

```
<Directory directory_name>
    Options -Indexes
</Directory>
```



## Exercice 2

- a) Reprendre l'exercice 2 et reconfigurez le serveur pour qu'il écoute sur le port 80 sur votre interface.
  - b) Quel est le Module Multi-Processus (MPM) utilisé actuellement ?
  - c) Combien de processus enfants y a-t-il actuellement ?
  - d) Configurez-le pour:
    - qu'il lance 7 processus enfant en mode 'prefork' au démarrage et qu'il s'assure qu'il en reste toujours au moins 3 en réserve. En cas de montée en charge, 20 processus enfants maximum seront créés pour satisfaire les requêtes.  
Lorsque le serveur aura satisfait toutes les requêtes de cette montée, il se stabilisera avec 10 processus enfants.
- Testez tout cela, à l'aide d'outils adéquats.



## Exercice 2 (Suite)

- que la requête `http://www.mysite.be/pamauthor` présente la page `/usr/share/doc/pam/html/sag-author.html`
- qu'il donne la possibilité aux utilisateurs jean, louis et bernard de disposer d'un site perso dont la page d'accueil puisse être `index.html` ou `accueil.html` (inventez leur contenu).  
Le site perso de bernard ne sera pas accessible pour le moment.
- qu'un index des entrées du dossier 'cours' soit présenté lors de la requête `http://www.mysite.be/cours`

e) Testez vos configurations.



# Protection des sites web

- **Objectifs**

Bloquer l'accès à une partie d'un site en limitant sa consultation à quelques utilisateurs privilégiés ou à quelques machines.

- Accéder à une porte d'administration du site (login membres ...)
- Verrouiller une partie du site en construction.
- Permettre des accès seulement à un ou plusieurs réseaux de machines.
- ...



# Protection des sites web

- **Contrôle sur l'origine du client**

La directive `Require [not]` est utilisée au cours de la phase d'**autorisation** pour accorder ou refuser l'accès à une ressource à un client.

## Exemples

`Require all granted` → l'accès est autorisé sans restriction.

`Require all denied` → l'accès est systématiquement refusé.

- ➥ Le fournisseur d'autorisation **all** est pris en charge par le module `mod_authz_core`

Plus d'info [https://httpd.apache.org/docs/2.4/fr/mod/mod\\_authz\\_core.html](https://httpd.apache.org/docs/2.4/fr/mod/mod_authz_core.html)



# Protection des sites web

- **Contrôle sur l'origine du client (suite)**

## Autres exemples

Require **ip** 10/255.0.0.0 172.20/16 192.168.2.4

→ Accès si l'@Ip du client fait partie d'une des tranches spécifiées.

Require **not ip** 10 172.20/16 192.168.2.4

→ Refus si l'@Ip du client fait partie d'une des tranches spécifiées.

Require **host** www.example.org

→ Les hôtes dont les noms correspondent ou se terminent par la chaîne spécifiée se verront accorder l'accès.

Require **not host** .net example.edu

→ Les hôtes dont les noms correspondent ou se terminent par la chaîne spécifiée se verront refuser l'accès.

☞ Les fournisseurs d'autorisation **ip** et **host** sont pris en charge par le module `mod_authz_host`

Plus d'info [https://httpd.apache.org/docs/2.4/fr/mod/mod\\_authz\\_host.html](https://httpd.apache.org/docs/2.4/fr/mod/mod_authz_host.html)



# Protection des sites web

- **Contrôle sur l'origine du client (suite)**

## L'intervant DNS

Require **host** www.example.org

Require **not host** .net example.edu

- Le fournisseur d'autorisation **host** va effectuer une double recherche DNS sur l'adresse IP du client :
  - une recherche DNS inverse sur l'adresse IP source pour trouver le nom associé
  - puis une recherche DNS directe sur le nom renseigné dans la directive pour trouver l'adresse IP originale.

L'accès ne sera accordé que si les recherches DNS inverse et directe sont cohérentes.

## Tutoriel sur le contrôle d'accès

<https://httpd.apache.org/docs/2.4/fr/howto/access.html>



# Protection des sites web

- **Contrôle sur l'origine du client (suite)**

Les conteneurs d'autorisation: <RequireAll>, <RequireAny> et  
<RequireNone>

Elles sont prises en charge par le module mod\_authz\_core.

```
<RequireAll>
    Require ...
    Require ...
    ...
</Require>
```

Si toutes les directives d'autorisation sont vraies alors l'accès est autorisé.

```
<RequireAny>
    Require ...
    Require ...
    ...
</Require>
```

Si au moins une directive d'autorisation est vraie alors l'accès est autorisé.

```
<RequireNone>
    Require ...
    Require ...
    ...
</Require>
```

Si au moins une directive d'autorisation est vraie alors l'accès est refusé.

- Les directives d'autorisation inversées (not) sont interdites dans les directives <RequireAny> et <RequireNone>.

## Plus d'info

[https://httpd.apache.org/docs/2.4/fr/mod/mod\\_authz\\_core.html#requireall](https://httpd.apache.org/docs/2.4/fr/mod/mod_authz_core.html#requireall)



# Protection des sites web

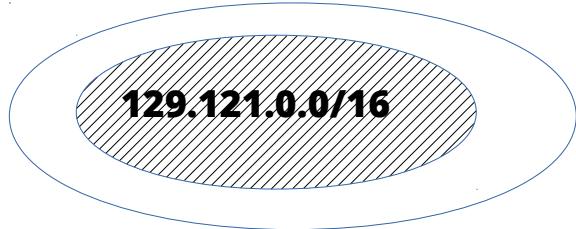
- Contrôle sur l'origine du client (suite)

## Exemples

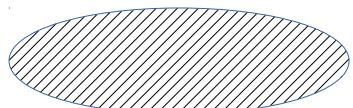
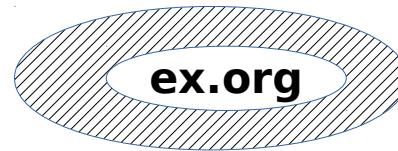
Require ip 129.121 →



Require not ip 129.121 →

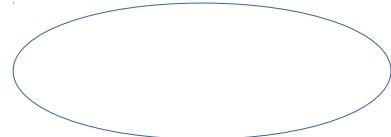


Require host ex.org →



← Require all denied

Require all granted →



# Protection des sites web

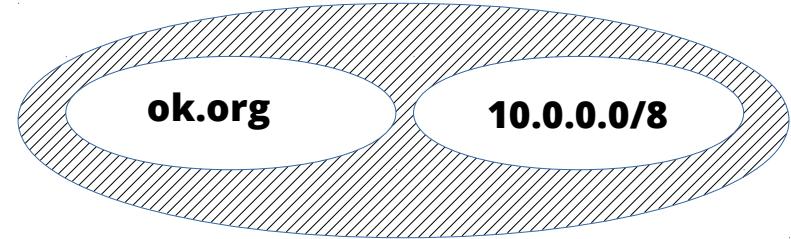
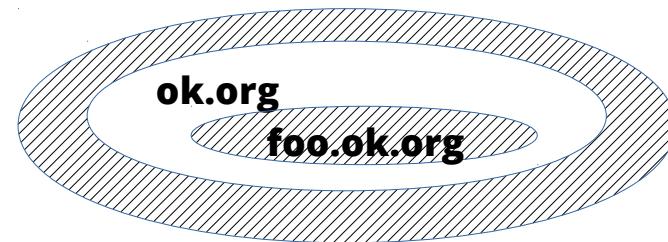
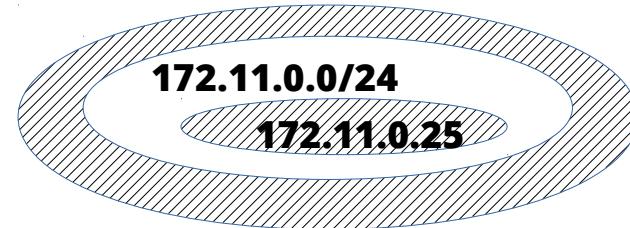
- Contrôle sur l'origine du client (suite)

## Exemples

```
<RequireAll>
    Require ip 172.11.0.0/24
    Require not ip 172.11.0.25
</Require>
```

```
<RequireAll>
    Require host ok.org
    Require not host foo.ok.org
</Require>
```

```
<RequireAny>
    Require host ok.org
    Require ip 10
</Require>
```

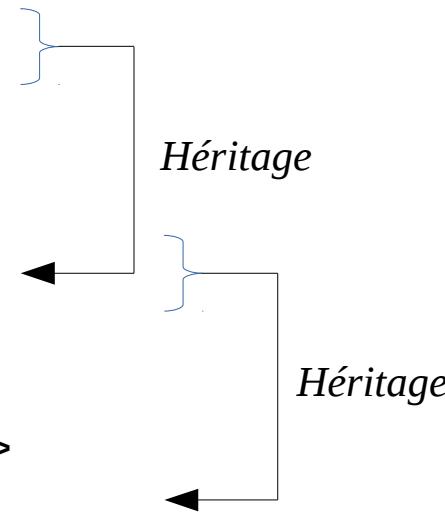


# Protection des sites web

- **Le conteneur <Directory>**

Il contient des directives qui ne s'appliquent qu'à celui-ci.

```
<Directory />
  ...
</Directory>
<Directory /var/www/html>
  ...
    } Ajouts ou redéfinitions
</Directory>
<Directory /var/www/html/private>
  ...
    } Ajouts ou redéfinitions
</Directory>
```



# Protection des sites web

- **Contrôle par authentification**

## L'authentification HTTP de base:

- Gérée par les modules `mod_auth_basic` (type : Basic ou Digest), `mod_authz_user` (**autorisation**) et `mod_authn_file` (**authentification**).
- Les "users/passwords" sont stockés dans un fichier texte.
- L'utilitaire `htpasswd` permet de créer et mettre à jour ce fichier.
- Les mots de passe de ce fichier sont cryptés à l'aide de la fonction `crypt`.
- Cette méthode est la plus simple mais la moins sécurisée car les identifiants sont transmis chiffrés en 'Base64' (très facilement déchiffrables). A n'utiliser que sous https !!!

## Exemple:

```
# htpasswd -c /var/www/securite/userfile toto → création (en dehors du site...  
                                     sécurité oblige ! )
```

```
# cat /var/www/securite/userfile  
toto:cnVPtfAz2xw60
```

```
# htpasswd -b /var/www/securite/userfile albert secret → ajout avec mdp
```

- ☞ -m → pour chiffrer le mdp en MD5      -p → pas de chiffrement
- s → pour chiffrer le mdp en SHA



# Protection des sites web

- **Contrôle par authentification**

## L'authentification HTTP de base: (suite)

Activation de l'autorisation:

### Exemple

```
<Directory rep_a_proTEGER>
    AuthName "Domaine de test" → identificateur d'autorisation
    AuthType Basic → type d'authentification (Basic ou Digest)
```

```
AuthUserFile /var/www/securite/userfile → (1)
```

```
    Require valid-user → (2)
</Directory>
```

<sup>(1)</sup> Fichier contenant la liste des utilisateurs qui ont le droit de s'authentifier (mod\_authn\_file). C'est le **fournisseur d'authentification**.

<sup>(2)</sup> Liste des utilisateurs qui peuvent accéder aux ressources après s'être authentifier correctement (mod\_authz\_user).

Avec Require valid-user: tous les utilisateurs repris dans la liste<sup>(1)</sup>.

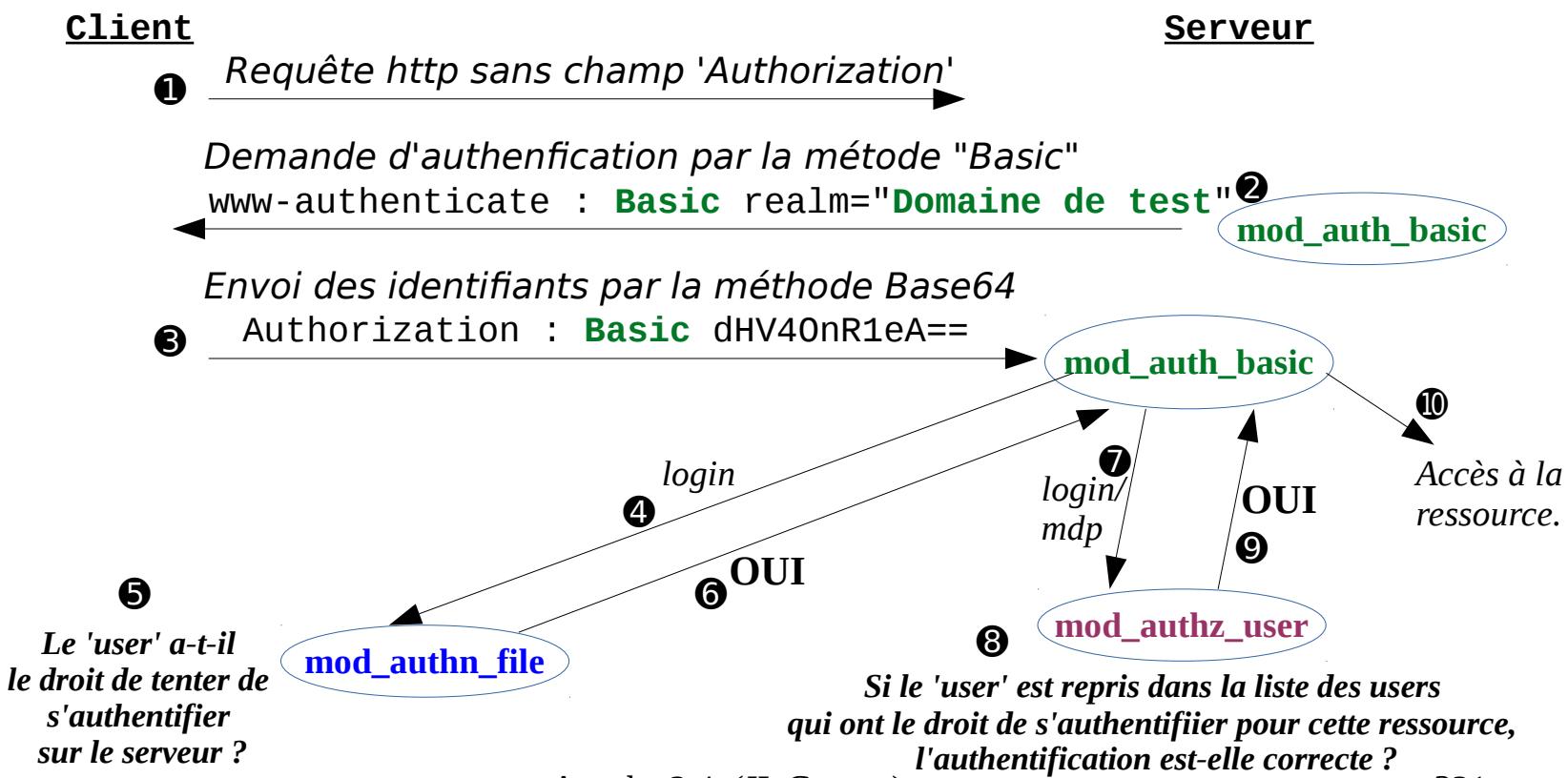


# Protection des sites web

- Contrôle par authentification

L'authentification HTTP de base: (suite)

**Synoptique:**



# Protection des sites web

- **Contrôle par authentification**

- L'authentification HTTP de base: (suite)

```
<Directory rep_a_proteger>
    AuthName "Domaine de test"
    AuthType Basic
    AuthUserFile /var/www/securite/userfile
    Require user toto albert
</Directory>
```

- ☞ Ici, seuls 'toto' et 'albert' repris dans 'userfile' ont le droit d'accéder aux ressources du répertoire moyennant une authentification valide.



# Protection des sites web

- **Contrôle par authentification**

## L'authentification HTTP de base: (suite)

Soit un fichier texte /var/www/securite/groupfile contenant:

admins:toto albert

friends:toto linda

associates:bruno      alors:

```
<Directory rep_a_proTEGER>
    AuthName "Domaine de test"
    AuthType Basic
    AuthUserFile /var/www/securite/userfile
    AuthGroupFile /var/www/securite/groupfile
    Require group admins friends
</Directory>
```

- ☞ Ici, seuls les membres des groupes 'admins' et 'friends' ont le droit d'accéder aux ressources du répertoire moyennant une authentification valide.

La directive **AuthUserFile** reste obligatoire car elle contient la liste des **users qui peuvent s'authentifier**.

La directive **Require** détermine **qui a l'autorisation** d'accéder aux ressources **après authentification** réussie).



# Protection des sites web

- **Autres types d'authentification**

- Par digest: le mot de passe ne circule plus en clair mais tous les navigateurs ne gèrent pas cette méthode.
- Par db: Unix DBM, MYSQL, Microsoft SQL Server, Oracle, Postgres ...
- Par serveurs d'authentification: Radius, Ldap, Kerberos, Samba, Nis ...

☞ Pour une liste à jour :

<http://httpd.apache.org/docs/2.4/fr/mod> (mot clé: *auth*)

☞ Pour un tuto sur l'authentification et autorisation :

<https://httpd.apache.org/docs/2.4/fr/howto/auth.html>



# Exercice 3

- Configurez Apache pour:
  - . rendre l'entrée d'un intranet (`http://www.mysite.be/intranet`) uniquement accessible par les machines de notre réseau local `10.103.0.0/16`.
  - . que les utilisateurs 'tux', 'bill', 'louis' et 'jean' puissent s'authentifier lorsqu'ils accèdent au site `http://www.mysite.be/beta`; en sachant que seuls 'tux' et 'bill' auront le droit d'accéder à ses ressources.
- Testez vos configurations.
- Expliquez le principe de l'authentification Basic.
- Expliquez le codage Base64.
- Testez avec tshark que la méthode d'authentification 'Basic' n'est pas sécurisée.
- Testez d'autres outils permettant de (dé)coder un message chiffré par la méthode Base64.



# Hébergement virtuel

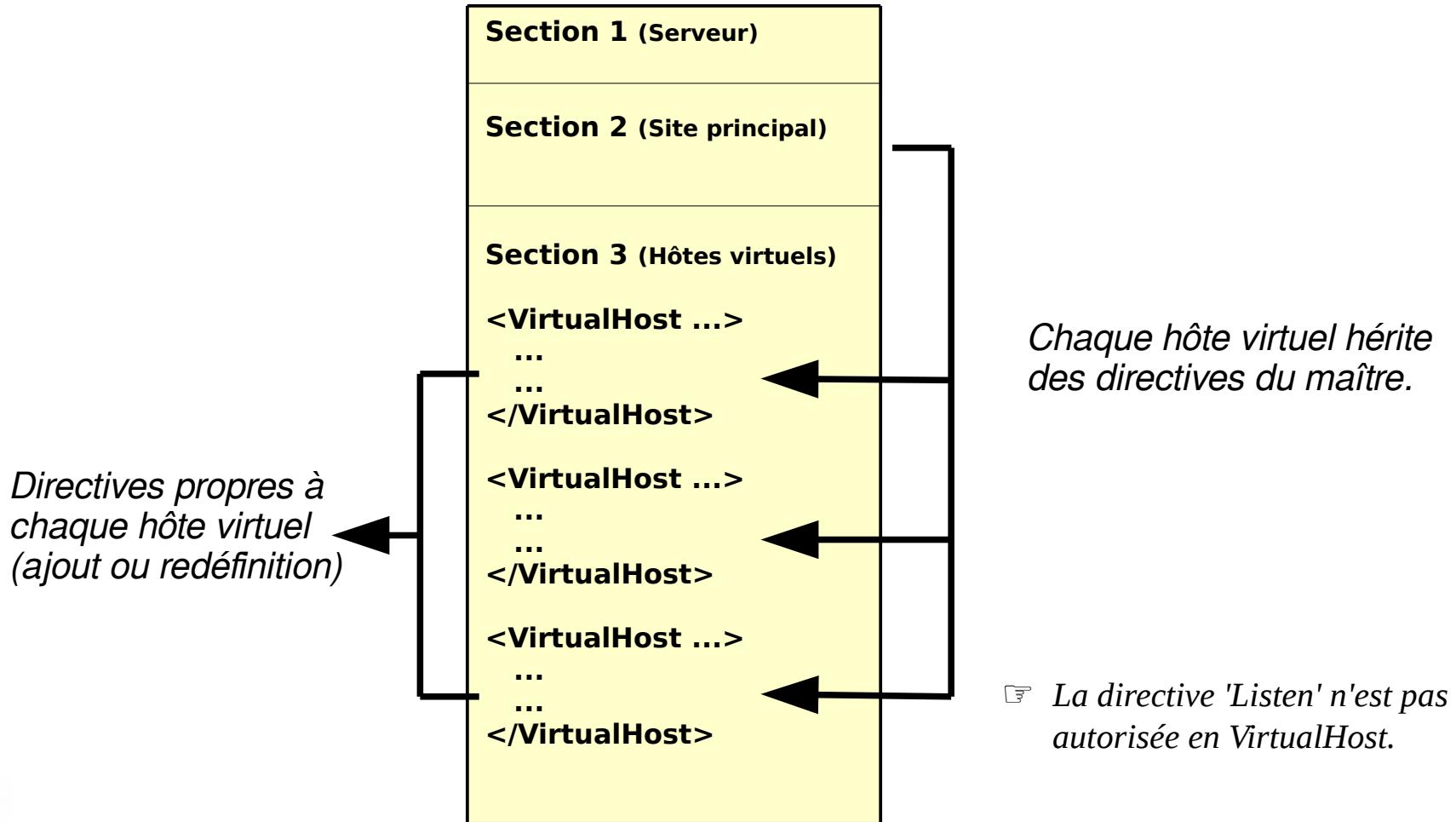
- **Introduction**

- Pour gérer plusieurs sites web sur une seule machine serveur.
- 2 méthodes:
  - a) Hébergement virtuel par adresse IP.
  - b) Hébergement virtuel par nom.



# Hébergement virtuel

- **Principe**



# Hébergement virtuel

- **Hébergement par IP**

- Les sites sont identifiés par l'IP sur laquelle arrivent les requêtes.
- Une IP dédiacée à un seul site
- Besoin de plusieurs interfaces physiques → nécessité d'alias.
- De moins en moins utilisé (beaucoup d'ip monopolisées).
- Les clients non HTTP1.1 (rare) ont néanmoins besoin de ce type d'hébergement.



# Hébergement virtuel

- Hébergement par IP

## httpd.conf

```
ServerName www.mysite.be
DocumentRoot /var/www/html
```

CONNECT 192.168.1.1:80  
GET / HTTP/1.1

<http://192.168.1.1>

*On reçoit la page d'accueil du site principal.*

```
<VirtualHost 192.168.1.4>
  ServerName vhost1.mysite.be
  DocumentRoot /var/www/html/vhost1
</VirtualHost>
```

CONNECT 192.168.1.4:80  
GET / HTTP/1.1

<http://192.168.1.4>

*On reçoit la page d'accueil de vhost1.mysite.be*

```
<VirtualHost 192.168.1.5>
  ServerName vhost2.mysite.be
  DocumentRoot /var/www/html/vhost2
</VirtualHost>
```

CONNECT 192.168.1.5:80  
GET / HTTP/1.1

<http://vhost2.mysite.be>

*Après résolution du nom, on reçoit la page d'accueil de vhost2.mysite.be*



# Hébergement virtuel

- **Hébergement par IP**

Remarques:

- La création d'un alias sur une interface *iface* se fait par la création d'un fichier :

```
/etc/sysconfig/network-scripts/ifcfg-iface:0
...
IPADDR=nouvelle_ip
NETMASK=nouveau_masque
ONBOOT=yes
NAME=iface:0
DEVICE=iface:0
```



Numéro d'alias

- Pour créer d'autres alias alias sur cette même interface, il suffit de créer d'autres fichiers en incrémentant le numéro d'alias (...:**1**, ...:**2**, etc ) et en adaptant les valeurs des variables NAME et DEVICE en conséquence.
- Si une adresse IP peut être atteinte mais qu'aucun hôte n'est défini sur celle-ci, c'est le site principal qui répondra à la requête.



# Hébergement virtuel

- **Hébergement par nom**

- Une IP dédiacée à plusieurs sites.
- Les sites sont identifiés par le nom du serveur auquel s'adresse le client.
- Un nom par hôte virtuel.
- Exploitable à partir de HTTP1.1 qui véhicule le nom de l'hôte via une en-tête Host dans sa requête. Il permet ainsi d'identifier sa cible.
- Si le serveur utilise un hébergement par IP , l'en-tête Host est ignorée.
- Très utilisé chez les hébergeurs.



# Hébergement virtuel

- **Hébergement par nom**

## httpd.conf

```

ServerName www.mysite.be
DocumentRoot /var/www/html

<VirtualHost 192.168.1.1>
  ServerName www.mysite.be
  DocumentRoot /var/www/html
</VirtualHost>

<VirtualHost 192.168.1.1>
  ServerName vhost1.mysite.be
  DocumentRoot /var/www/html/vhost1
</VirtualHost>

<VirtualHost 192.168.1.1>
  ServerName vhost2.mysite.be
  DocumentRoot /var/www/html/vhost2
</VirtualHost>

<VirtualHost 192.168.1.2>
  ServerName vhost3.mysite.be
  DocumentRoot /var/www/html/vhost3
</VirtualHost>

<VirtualHost 192.168.1.2>
  ServerName vhost4.mysite.be
  DocumentRoot /var/www/html/vhost4
</VirtualHost>

```

CONNECT 192.168.1.1:80  
 GET / HTTP/1.1  
 Host : 192.168.1.1

CONNECT 192.168.1.1:80  
 GET / HTTP/1.1  
 Host : www.mysite.be

CONNECT 192.168.1.1:80  
 GET / HTTP/1.1  
 Host : vhost1.mysite.be

CONNECT 192.168.1.2:80  
 GET / HTTP/1.1  
 Host : vhost3.mysite.be

Apache 2.4 (JL Gouwy)

<http://192.168.1.1>

*On reçoit la page d'accueil du site principal.*

<http://www.mysite.be>

*Après résolution du nom, on reçoit la page d'accueil du site principal.*

<http://vhost1.mysite.be>

*Après résolution du nom, on reçoit la page d'accueil de vhost1.mysite.be*

<http://vhost3.mysite.be>

*Après résolution du nom, on reçoit la page d'accueil de vhost3.mysite.be*



# Hébergement virtuel

- **Hébergement par nom**

- A la lecture de `httpd.conf`, Apache crée une table contenant la liste des hôtes virtuels déclarés pour chaque Ip.

Ainsi, pour l'illustration précédente, la table se présente comme suit :

<b>Hébergement par nom sur l'IP</b>	<b>ServerName</b>		
192.168.1.1	www.mysite.be	vhost1.mysite.be	vhost2.mysite.be
192.168.1.2	vhost3.mysite.be	vhost4.mysite.be	

*Hôte virtuel primaire*

*Lorsque Apache reçoit une requête sur une Ip se trouvant dans la table, il recherche le ServerName qui correspond à l'en-tête Host de la requête ... Si aucune correspondance n'est trouvée, c'est l'hôte virtuel primaire qui sera sélectionné.*

*Ainsi, pour qu'il soit toujours accessible, on indique souvent le site maître en tant qu'hôte virtuel primaire dans l'entrée correspondant à son Ip*



## Exercice 4

Configurez Apache pour qu'il puisse en même temps gérer de l'hébergement par ip et par nom.

- 192.168.27.10 permettra d'héberger l'unique site `jean.mysite.be`
- 192.168.27.11 permettra d'héberger l'unique site `louis.mysite.be`
- 192.168.27.2 permettra d'héberger les sites `vh2.mysite.be` et `vh3.mysite.be`. Ce dernier ne sera accessible qu'aux users repris dans `/var/www/securite/pwd`.
- Le site maître sera toujours accessible par l'Ip 192.168.27.1 ou par `www.mysite.be`. Ce sera aussi le site par défaut si une requête arrive via une interface associée à aucun site.

Remarque: Les fonctionnalités des exercices précédents doivent toujours être opérationnelles.

Relancez Apache et testez sa configuration



# Les fichiers journaux

Pour véritablement gérer un serveur web, il est nécessaire de disposer d'un retour d'informations à propos de l'activité et des performances du serveur, ainsi que de tout problème qui pourrait survenir.

- **Le journal des erreurs** (Directives ErrorLog & LogLevel)

Directive ErrorLog : Le nom et la localisation du journal.

## Exemples

ErrorLog logs/errors-logs

Enregistrement vers un fichier particulier.

ErrorLog "| /usr/local/bin/erreurs\_httpd"

Traitements de l'erreur par un binaire.

ErrorLog syslog:user

Traitements de l'erreur par le daemon syslogd.



# Les fichiers journaux

- **Le journal des erreurs**

Directive LogLevel : Indique quels sont les messages à écrire dans le fichier journal.

Exemple      LogLevel warn

## Tableau des criticités en ordre croissant

emerg	Urgences - le serveur est inutilisable.
alert	Des mesures doivent être prises immédiatement.
crit	Conditions critiques (accès réseau impossible par ex.).
error	Erreurs dans les pages, les scripts.
warn	Avertissements (pages mal codées, erreurs non bloquantes dans un script...).
notice	Événement important mais normal.
info	Informations.
debug	Enregistre TOUT ce qui se passe sur le serveur.

- ☞ *Lorsqu'un niveau particulier est spécifié, les messages de tous les autres niveaux de criticité supérieure seront aussi enregistrés (ex. le niveau crit enregistre en plus les messages de niveau alert et emerg ).*



# Les fichiers journaux

- **Le journal des erreurs**

## Format

[Wed Oct 11 14:32:52 2015] [error] [client 127.0.0.1] client denied by server configuration:  
/export/home/live/ap/htdocs/test

*Date et l'heure du message.*

*Sévérité de l'erreur rapportée (directive LogLevel)*

*Adresse IP du client qui a généré l'erreur.*

*Le message proprement dit, qui indique dans ce cas que le serveur a été configuré pour interdire l'accès au client. Le serveur indique le chemin système du document requis (et non son chemin web).*



# Les fichiers journaux

- **Le journal des accès**

Directives :      CustomLog, LogFormat & SetEnvIf  
Modules:           mod\_log\_config & mod\_setenvif

Directive CustomLog : La localisation du journal.

## Exemple

```
CustomLog logs/access_log common
```



*Les logs seront enregistrés selon l'alias 'common' défini dans la directive LogFormat (voir ci-après).*



# Les fichiers journaux

- **Le journal des accès**

Directive LogFormat : Le formatage des records du journal.

## Exemple

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

*Enregistrement des entrées de journalisation selon le format "Common Log Format" (CLF). Ce format standard peut être produit par de nombreux serveurs web différents et lu par de nombreux programmes d'analyse de journaux.*

127.0.0.1 - frank [10/Oct/2015:13:55:36 -0700] "GET /apache\_pb.gif HTTP/1.0" 200 2326

(%h)  
Adresse IP du client qui a envoyé la requête.

- (%l)  
Information non disponible.

(%u)  
Identifiant de la personne qui a demandé le document, issu d'une authentification HTTP.  
(tiret si absent)

(%t)  
L'heure à laquelle la requête a été reçue.

(\"%r\")  
Requête du client

(%>s)  
Code de statut que le serveur retourne au client.

(%b)  
Taille de l'objet retourné, entêtes non compris. (tiret si aucun contenu retourné).



# Les fichiers journaux

- **Le journal des accès**

- Autres types de journaux d'accès possibles :

- Journalisation combinée.
    - Journalisation multiple.
    - Journalisation conditionnelle.

- Autres tuning possibles :

- Rotation des journaux via des journaux redirigés.

- **Les hôtes virtuels**

- Même journal pour tous les logs de tous les hôtes virtuels.
  - Un journal séparé pour chaque hôte virtuel.
  - Un journal unique mais pouvant être parser via un programme tel que split-logfile.

Plus d'info : <https://httpd.apache.org/docs/2.4/fr/logs.html>



# Exercice 5

Continuez la configuration d'Apache pour qu'il réponde aux exigences suivantes :

	<b>Site maître</b>	<b>Sites virtuels</b>
<b>Journal des erreurs</b>		
Localisation	<b>logs/error_log</b>	<b>log/error_log</b>
Niveau de criticité	<b>debug</b>	<b>warn</b>
<b>Journal des accès</b>		
Localisation	<b>logs/access_log</b>	<b>logs/access_log.prefixe</b>
Format	<b>combined</b>	<b>common</b>

Qu'est-ce que le format combiné ? → Voir l'aide sur le site d'Apache

Testez votre nouvelle configuration.



# Les certificats

- **Objectifs**

- Pour pallier aux faiblesses du protocole http dans lequel rien n'est chiffré (y compris les échanges login/password)
- Pour pallier à la lacune les cryptosystèmes asymétriques où, lors du partage de la clé publique, rien ne garantit que la clé est bien celle de l'utilisateur à qui elle est associée.
- Ainsi, un certificat pourra prouver qu'une clé publique appartient bien à la personne qui prétend en être le propriétaire c'est-à-dire qui détient la clé privée correspondante.

Règle de **non-répudiation**



# Les certificats

- **SSL**

- Secure Socket Layer
- TLS : Transport Layer Security (nouveau standard)
- Ensemble de bibliothèques contenant tous les algorithmes permettant de chiffrer une communication:
  - . sans certificat (ex. ssh)
  - . avec certificat (ex. https)
- Exemple:

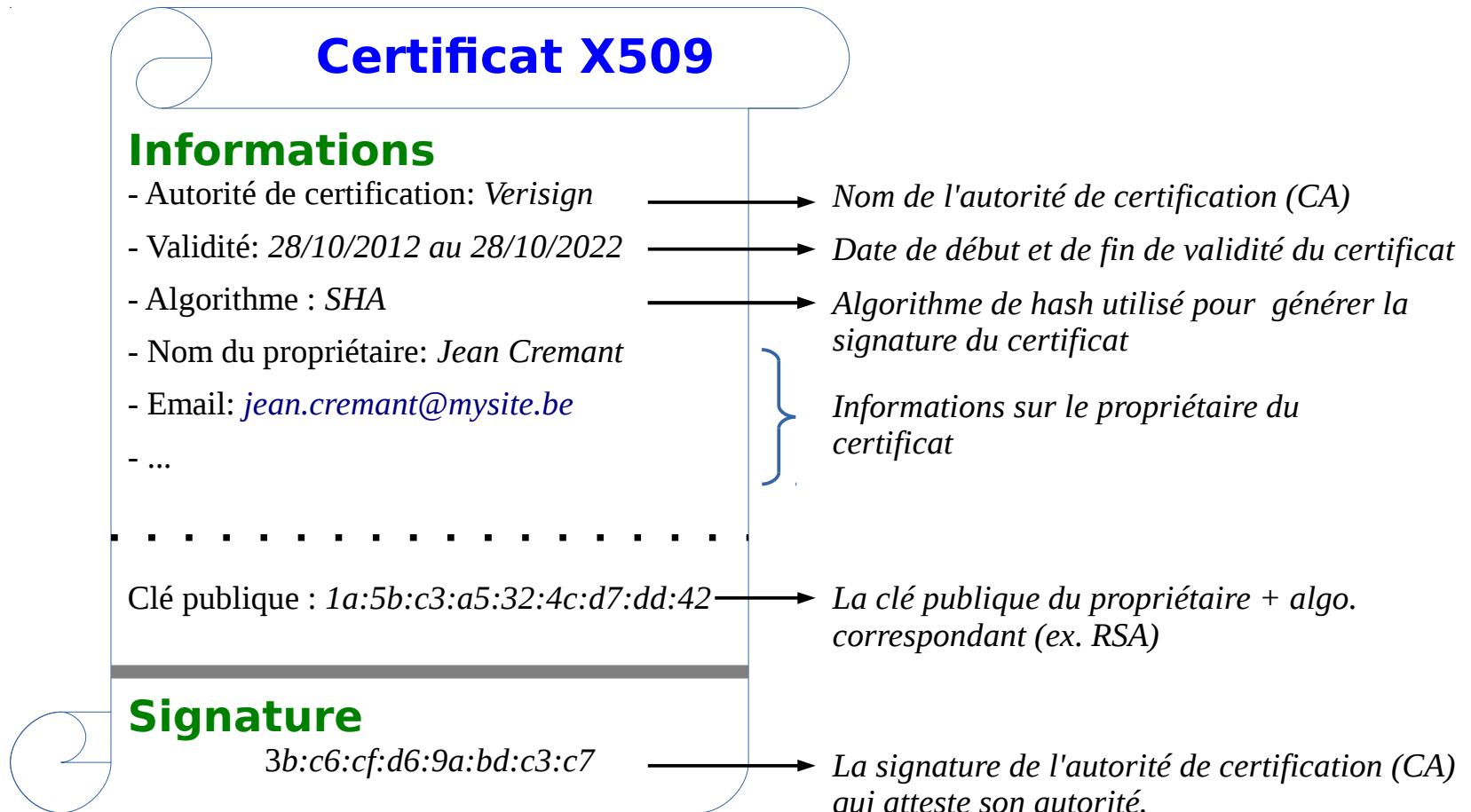
`https://secure.mysite.be`

Permet d'ouvrir une connexion vers le port `tcp 443` afin de négocier et d'entamer une communication `http` sécurisée avec le serveur.



# Les certificats

- **Structure générale (rfc5280)**



# Les certificats

- **Fonctionnement**

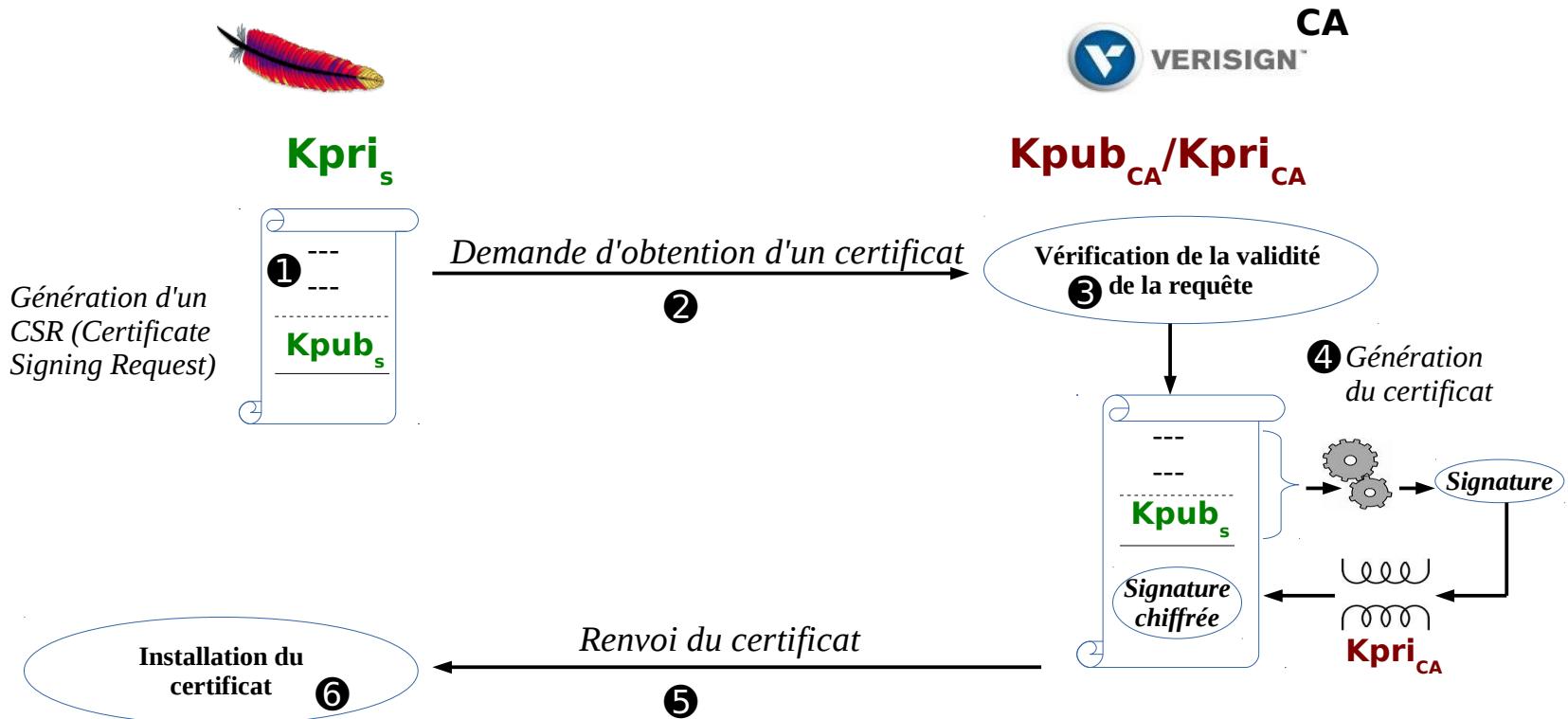
Trois intervenants :

- Un client Web
- Un serveur Web
- Une autorité de certification (CA)



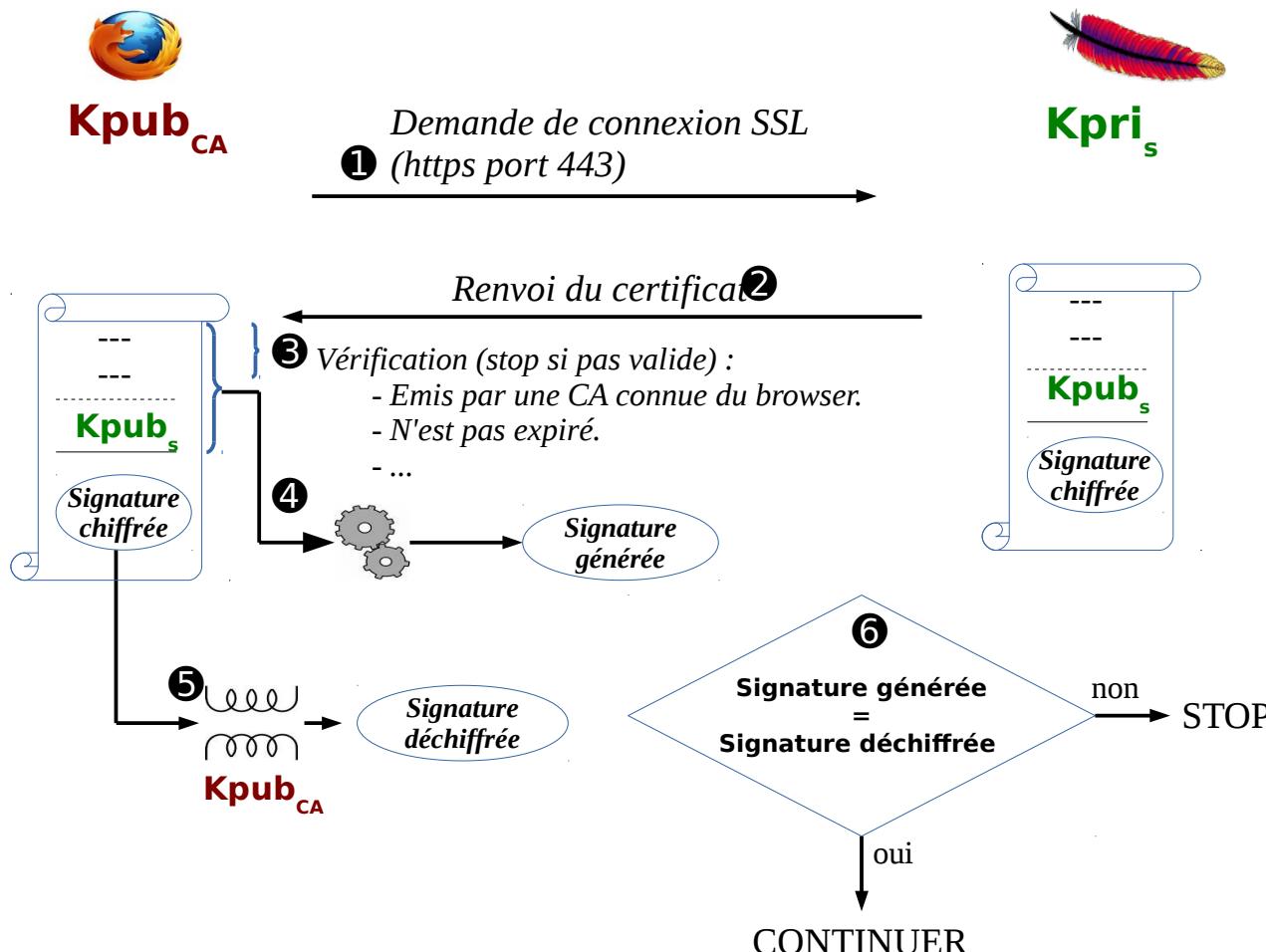
# Les certificats

- Fonctionnement : Obtention d'un certificat**



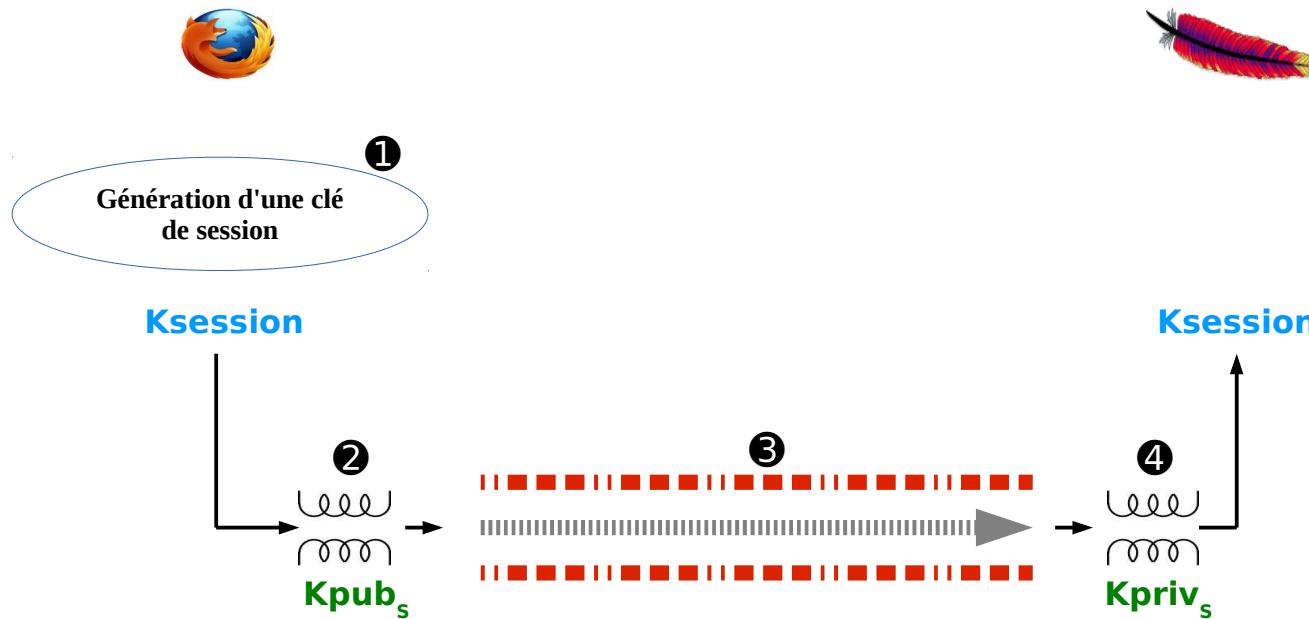
# Les certificats

- Fonctionnement : Connexion SSL



# Les certificats

- Fonctionnement : Génération d'une clé de chiffrement symétrique



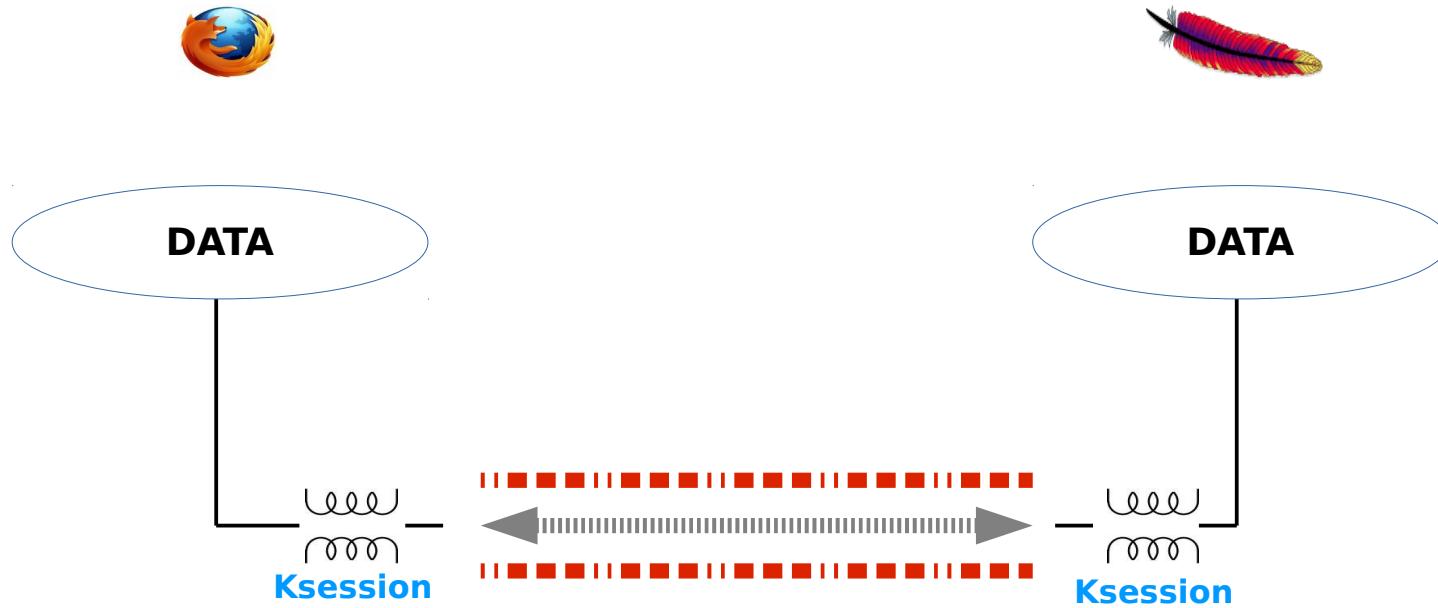
*L'échange de la clé de session connaît d'autres variantes en fonction de la version du protocole SSL utilisé (SSL v2, SSL v3) ...*

Plus d'info sur un échange SSL: [http://httpd.apache.org/docs/trunk/fr/ssl/ssl\\_intro.html](http://httpd.apache.org/docs/trunk/fr/ssl/ssl_intro.html)



# Les certificats

- Fonctionnement : Communication SSL



# Les certificats

- **Question de confiance**

- Lorsqu'un client désire communiquer en SSL avec un serveur, il lui suffit de se procurer le certificat de ce dernier.
- Ce certificat valide l'identité du serveur.
- Ici, la règle de non-répudiation repose donc sur la confiance envers l'autorité de certification.



# Les certificats

- **Signature**

Les certificats auto-signés:

- Signés par un serveur local.
- Pour garantir la confidentialité des échanges au sein d'un intranet.

Les certificats signés par un CA:

- Pour garantir la confidentialité des échanges au sein de l'internet.
- Le certificateur tiers permet d'assurer à l'utilisateur que le certificat appartient bien à l'organisation à laquelle il est déclaré appartenir.



# Les certificats

- **Implémentation**

## Implémentation d'un certificat auto-signé

Nous travaillerons ici avec un certificat auto-signé pour ne pas devoir l'enregistrer auprès d'une autorité de certification extérieure... (quelques centaines d'euros/an).

Etape 1: Installer OpenSSL et le module ssl d'Apache

```
# yum install openssl mod_ssl
```

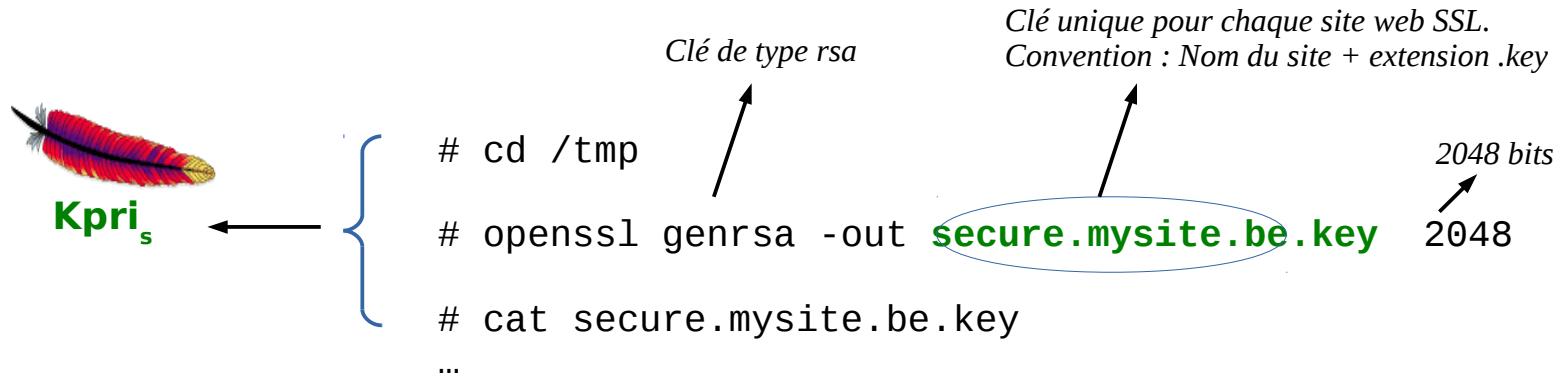


# Les certificats

- **Implémentation**

## Implémentation d'un certificat auto-signé (suite)

Etape 2: Création de la clé privée du serveur

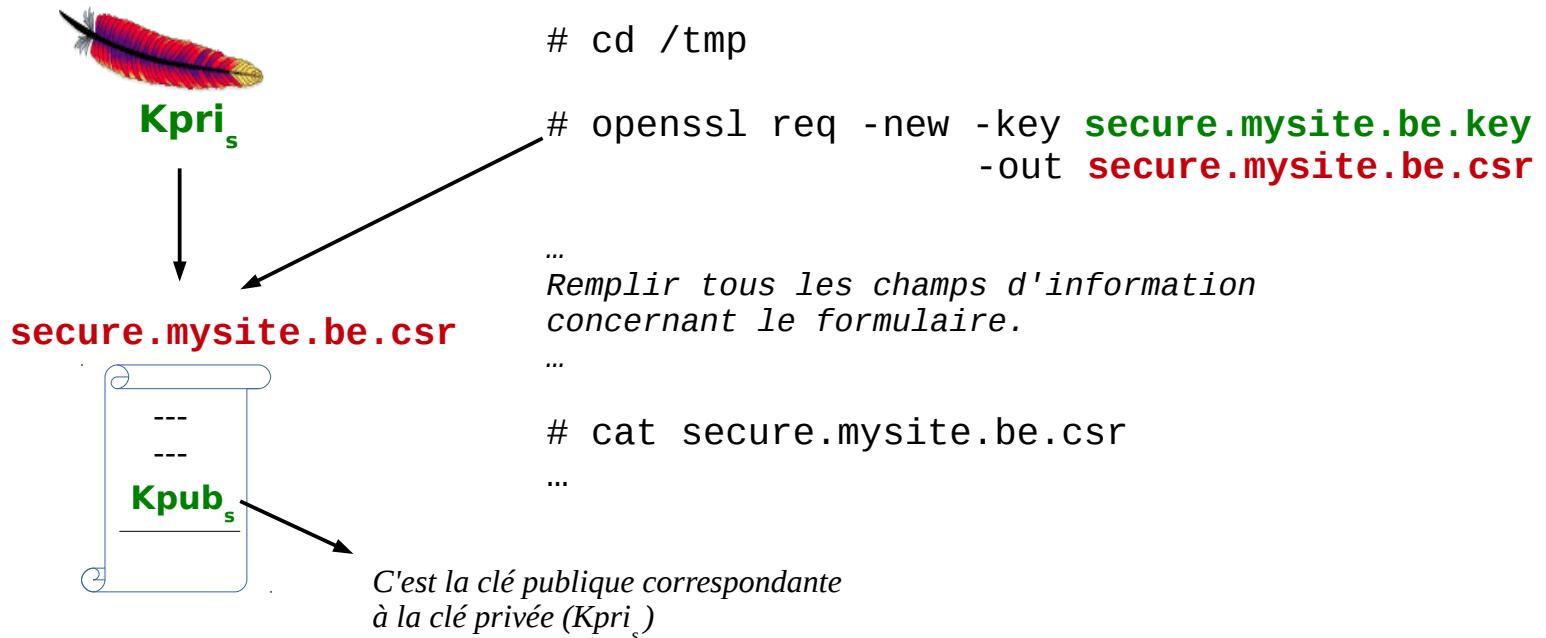


# Les certificats

- **Implémentation**

## Implémentation d'un certificat auto-signé (suite)

Etape 3: Création du CSR (Certificate Signing Request)

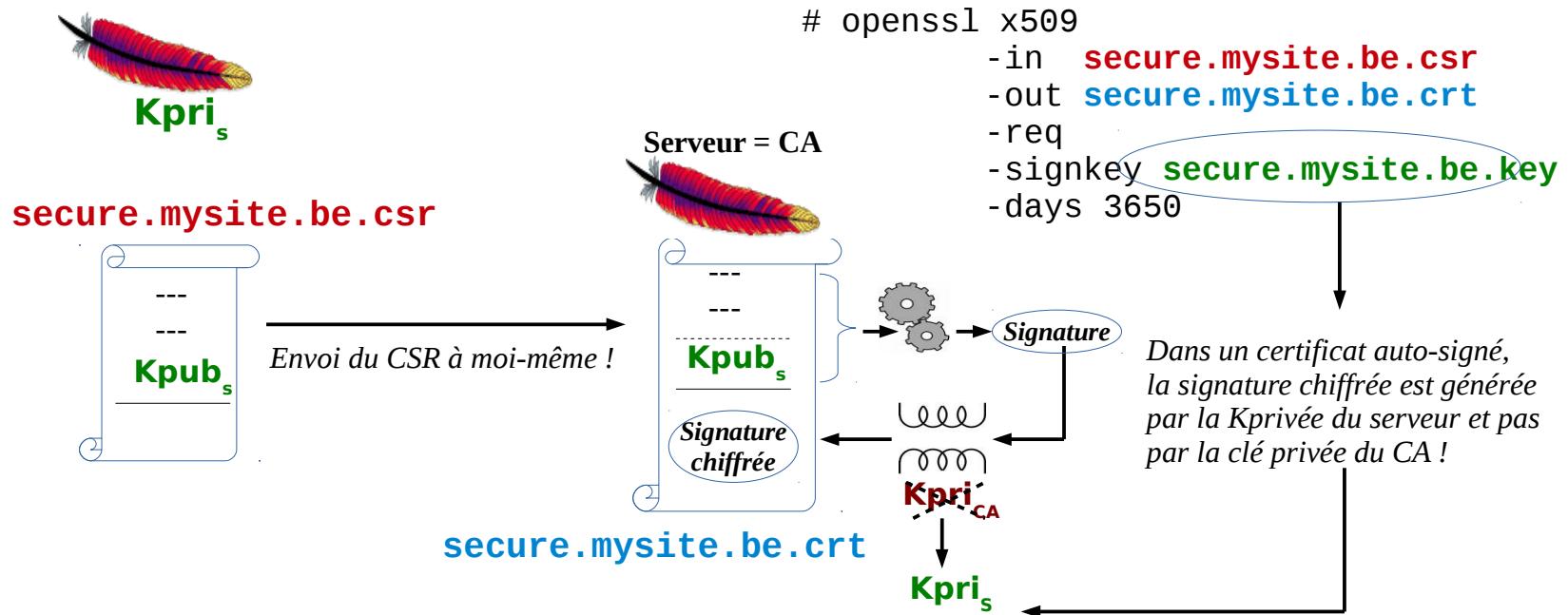


# Les certificats

- **Implémentation**

## Implémentation d'un certificat auto-signé (suite)

Etape 4: Création du certificat

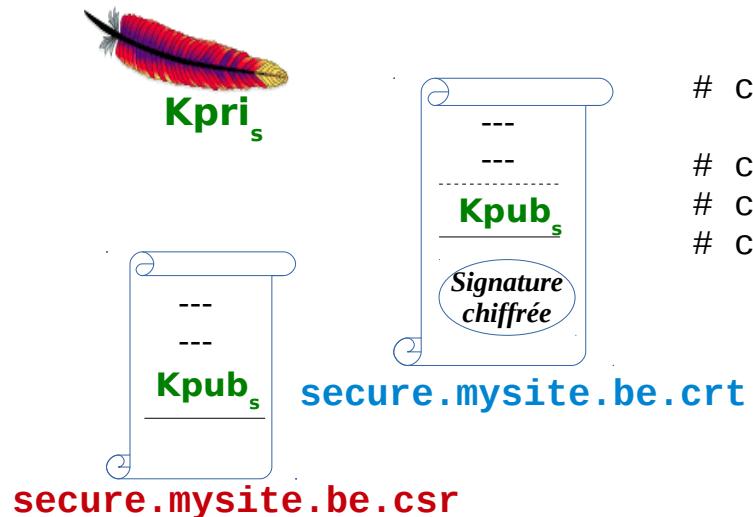


# Les certificats

- **Implémentation**

## Implémentation d'un certificat auto-signé (suite)

Etape 5: Copie des clés et des certificats au bon endroit



```
# cd /tmp  
# cp secure.mysite.be.crt /etc/pki/tls/certs  
# cp secure.mysite.be.csr /etc/pki/tls/private  
# cp secure.mysite.be.key /etc/pki/tls/private
```

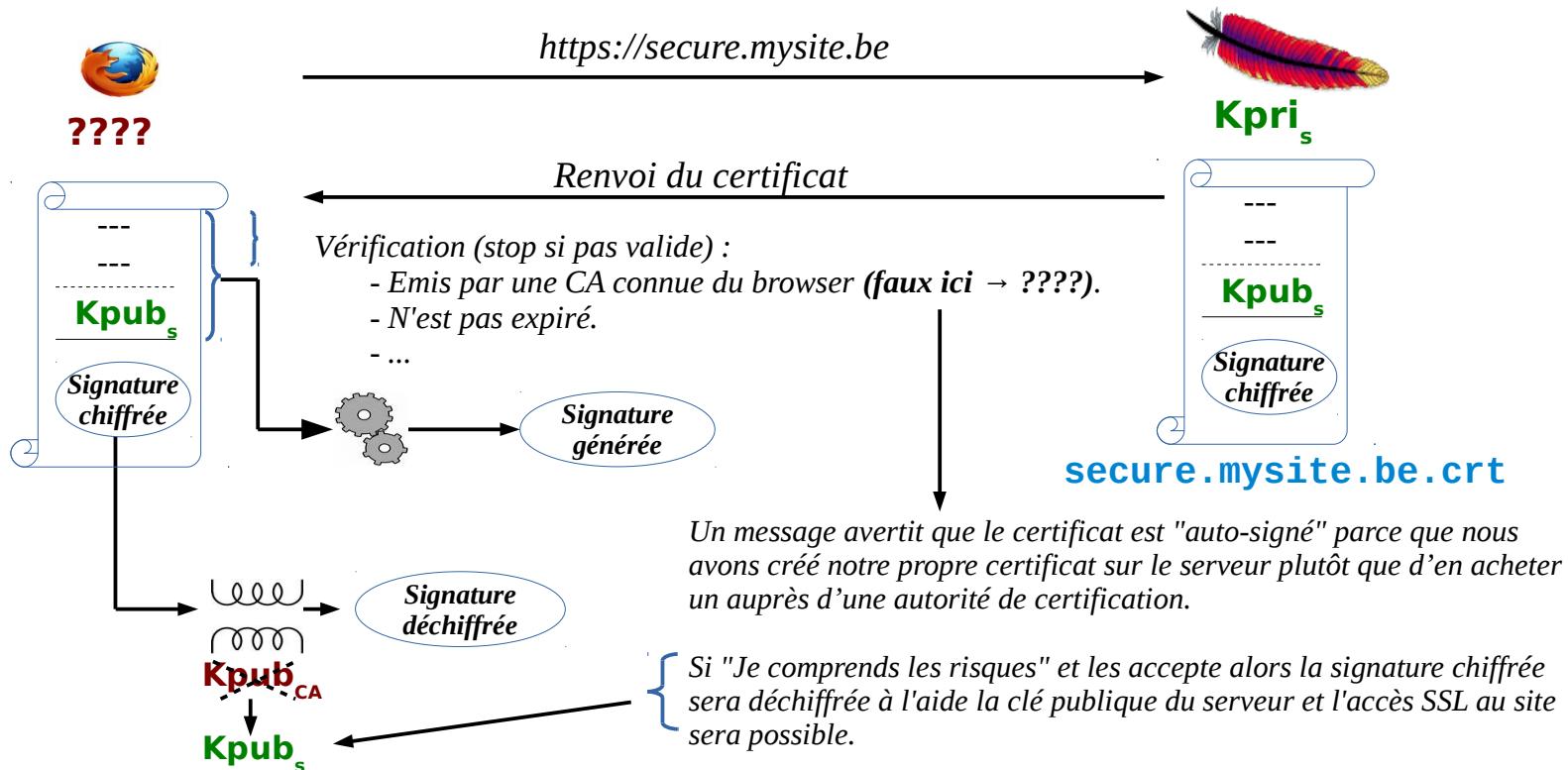


# Les certificats

- **Implémentation**

## Implémentation d'un certificat auto-signé (suite)

### Etape 6: Connexion SSL



# Les certificats

- **Implémentation**

## Remarques

- Pour ne pas devoir "Comprendre les risques" lors de chaque connexion, il faut importer le certificat (.crt) du serveur dans le navigateur client.
- Lire le contenu d'un certificat: # openssl x509 -noout -text -in mycertif.crt
- Soumettre son .csr à un CA:
  - . Choisir un CA reconnu par tous les navigateurs (ex. Verisign). On disposera alors de sa clé publique.
  - . Se connecter sur la page d'enregistrement du CA.
  - . Copier/Coller son .csr dans le formulaire de cette page.
  - . Répondre à quelques questions supplémentaires.
  - . Réaliser le paiement.
  - . Le certificat sera envoyé par mail dans les jours qui suivent.



# Les certificats

- **Implémentation**

## Configuration d'Apache

- Soit:
  - . https://secure.mysite.be sur 192.168.1.100 (hôte virtuel)
  - . Page d'accueil dans /var/www/html/secure
- Configurer /etc/httpd/conf.d/01-ssl.conf

```
<VirtualHost 192.168.1.100:443>
    ServerName secure.mysite.be
    DocumentRoot /var/www/html/secure
    SSLEngine on   → Pour spécifier que SSL doit être utilisé pour un hôte virtuel et pas pour le serveur.
    SSLCertificateFile /etc/pki/tls/certs/secure.mysite.be.crt
                    → Nom complet du certificat pour ce site.
    SSLCertificateKeyFile /etc/pki/tls/private/secure.mysite.be.key
                    → Nom complet de la clé privée pour ce site.
</VirtualHost>
```



# Les certificats

- **Implémentation**

## Configuration d'Apache

- Vérifier dans /etc/httpd/conf.modules.d/00-ssl.conf

...  
LoadModule ssl\_module modules/mod\_ssl.so → Nécessaire aux certificats.

- Configurer /etc/httpd/conf.d/01-ssl.conf

...  
Listen 443 → Apache écoutera également sur le port 443 de toutes ses interfaces.

...  
SSLPassPhraseDialog builtin → Une passphrase éventuelle sera demandée sur l'entrée standard.

SSLSessionCacheTimeout 300 → Timeout pour les sessions SSL (300 secondes).

...

→ Toutes ces directives générales seront exécutées au démarrage d'Apache (ne rien changer).

Certaines d'entre elles sont redéfinies dans l'hôte virtuel (voir ci-avant)



# Les certificats

- **La redirection SSL**

## Pourquoi ?

- Obliger une navigation sécurisée.
- Un des facteurs de référencement de Google.

## Exemple 1

```
<VirtualHost 192.168.1.1:80>
    ServerName www.mysite.be
    DocumentRoot /var/www/html
    Redirect permanent /secure https://www.mysite.be/secure
</VirtualHost>
```

```
<VirtualHost 192.168.1.1:443>
    ServerName www.mysite.be
    DocumentRoot /var/www/html/secure
    SSLEngine on
    ...
</VirtualHost>
```

*Toute requête à destination de la ressource /secure se trouvant dans www.mysite.be sera redirigée vers la page d'accueil se trouvant dans /var/www/html/secure. A partir de ce moment, la navigation SSL sera initiée.*

*La redirection est 'permanente' c'est-à-dire que le client recevra un code de retour 301 lui indiquant que la ressource a été déplacée définitivement (il devrait donc accéder à cette URL via le nom www.mysite.be/secure ).*



# Les certificats

- **La redirection SSL**

## Exemple 2

```
<VirtualHost 192.168.1.1:80>
    ServerName www.mysite.be
    DocumentRoot /var/www/html
    Redirect permanent / https://www.mysite.be
</VirtualHost>

<VirtualHost 192.168.1.1:443>
    ServerName www.mysite.be
    DocumentRoot /var/www/html
    SSLEngine on
    ...
</VirtualHost>
```

*Toute requête à destination de n'importe quelle ressource du site sera initiée en SSL.*



# Exercice 6

Continuez la configuration d'Apache pour que:

- la page d'accueil de `www.mysite.be` offre en plus la possibilité d'accéder à un secret via un lien "Cliquez ici pour voir le secret".
- un clic sur ce lien nous redirige vers la page d'accueil de `secure.mysite.be` d'un hôte virtuel SSL hébergé par Ip sur `192.168.27.100`. Cette page sera enregistrée dans `/var/www/html/websecure` et contiendra le secret.
  - . Importez le certificat dans Firefox de façon à ne plus avoir à "Comprendre les risques" lors de l'accès au site SSL.
  - . Tracez à l'aide d'un renifleur toute la conversation de `www.mysite.be` jusqu'à `secure.mysite.be`.
- les requêtes `http://vh3.mysite.be` et `https://vh3.mysite.be` nous offrent, de manière sécurisée, la page d'accueil du site.
  - . générez un nouveau certificat auto-signé pour ce site
  - . les 2 requêtes seront envoyées sur l'Ip `192.168.27.2`
  - . vérifiez par les logs de `access_log.vh3` que:
    - `http://vh3.mysite.be` génère 2 logs. Pourquoi ?
    - `http://vh3.mysite.be` génère 1 log. Pourquoi ?



# Références

## WEBOGRAPHIE

<http://christian.caleca.free.fr/http>  
<http://www.linux-france.org/prj/edu/archinet/systeme>  
<http://www.commentcamarche.net/contents/crypto>  
<http://ww2.ac-creteil.fr/reseaux/systemes/linux/index.html>  
<http://httpd.apache.org/docs/2.4/>

## BIBLIOGRAPHIE

Guide de référence 'Apache 2' - JM CULOT - OEM Eyrolles

Apache 2.0 - Guide de l'administrateur Linux - C. AULDS - Eyrolles



## Aspects Réseaux sous Linux

# Introduction au firewalling



Jean-Louis Gouwy



# Plan

- La sécurité
- Les pare-feu (firewalls)
  - Objectifs
  - Types de pare-feu
- Pare-feu dit "filtrant"
- Pare-feu dit "proxy"
- Topologie
  - Pare-feu unique
  - Double pare-feu
- Netfilter & Iptables
  - Définition
  - Fonctionnalités
  - Filtrage statique
  - Filtrage dynamique
  - Traduction d'adresse
  - Journalisation



# Plan

- Les concepts d'IPtables
  - Tables, chaînes et règles
  - La table "FILTER"
  - La table "NAT"
  - La table "MANGLE"
  - Hook points
  - Quelques actions (cibles)
- Exercices
  - Exercice 1 (Firewall stateless)
  - Exercice 2 (Firewall statefull)



# La sécurité

- Une solution de sécurité comprend plusieurs éléments : mots de passe, chiffrement, firewall ...
- Il est important que la solution choisie soit complète, car une chaîne est aussi résistante que le plus faible de ses maillons.
- La sécurité doit être assurée vis à vis de l'extérieur mais également de l'intérieur car c'est là que la majorité des problèmes trouve son origine.
- Le système de sécurité d'une entreprise se compose de nombreux outils complémentaires, un seul ne suffit généralement pas pour satisfaire l'objectif de sécurité fixé.
- Exemple de mise en œuvre:

Chiffrement, signature électronique (certificats), authentification, antivirus, firewall et proxys, VPN, fichiers logs ...



# Les pare-feu (firewalls)

## Objectifs

- Contrôler et limiter l'accès entre plusieurs réseaux au moyen d'un équipement doté de plusieurs interfaces réseaux (réelles ou virtuelles).
- Un pare-feu agit comme un entonnoir par lequel toutes les machines d'un réseau doivent passer pour communiquer vers l'extérieur.

## Types de pare-feu

- Deux grands types de pare-feu sont souvent mis en œuvre:
  - pare-feu "filtrant" (ex. iptables et netfilter)
  - pare-feu "proxy" (ex. squid)
- Ils sont très souvent complémentaires car ils permettent **à eux deux** d'entreprendre des actions en fonction du contenu des paquets de n'importe quelle couche du modèle TCP/IP.



# Pare-feu dit "filtrant"

- Il fonctionne au niveau des couches "non-applicatives".
- Il contrôle le flux du paquet en fonction de l'ip source, de l'ip de destination, du n° de port source ...
- Il doit pouvoir exécuter des actions en fonction de filtres appliqués sur les paquets.

ex. Filtres sur:

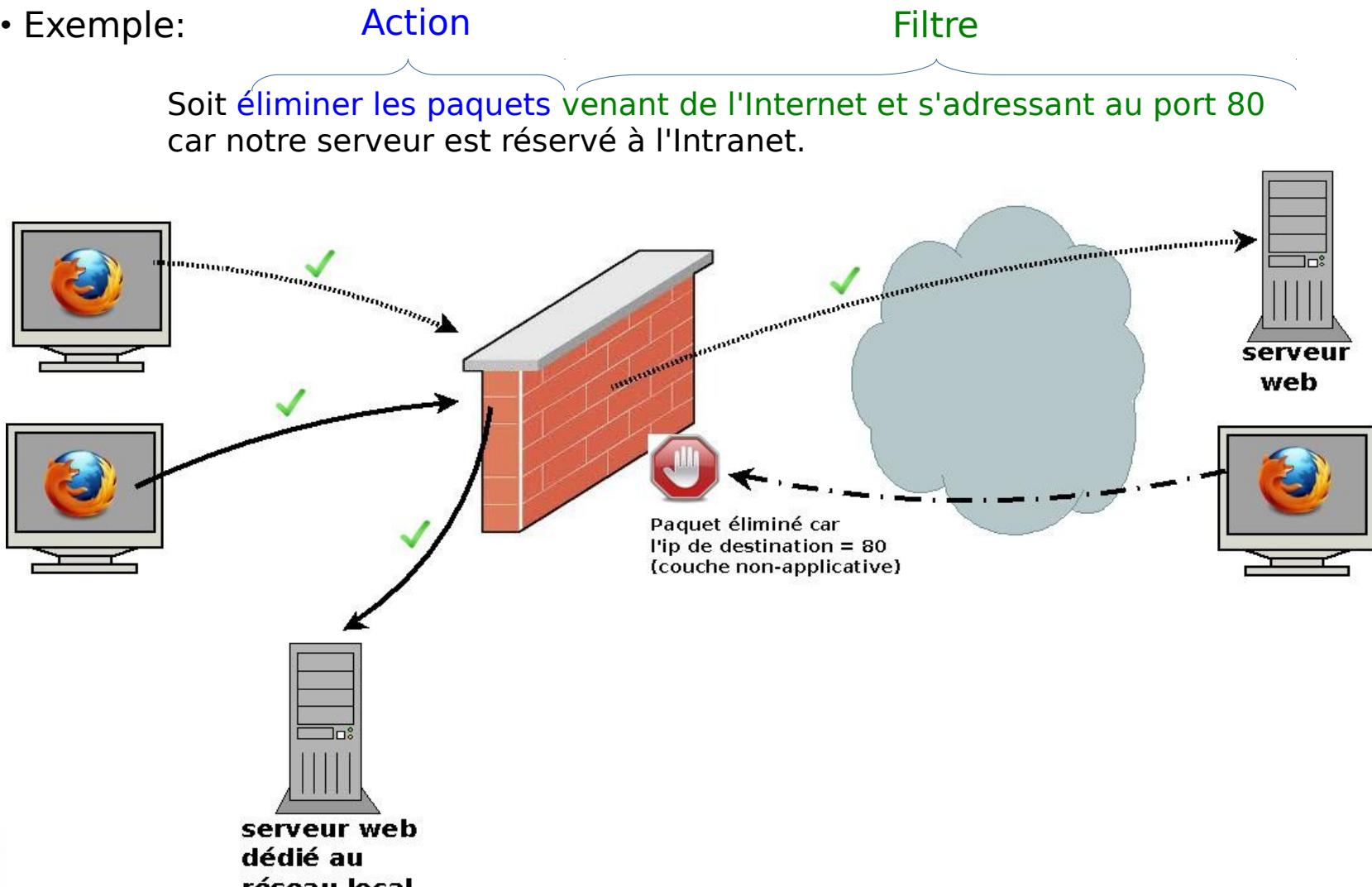
- le type de protocole (tcp, udp, icmp ...)
- le n° de port (pour tcp et udp)
- le type de paquet (syn/ack, icmp echo request ...)
- l'ip source et/ou de destination du paquet
- etc ...

- Il peut éventuellement inspecter le contenu des paquets pour vérifier leurs cohérence.
- Il doit pouvoir gérer l'état (*state*) d'une connexion.



# Pare-feu dit "filtrant" (suite)

- Exemple:



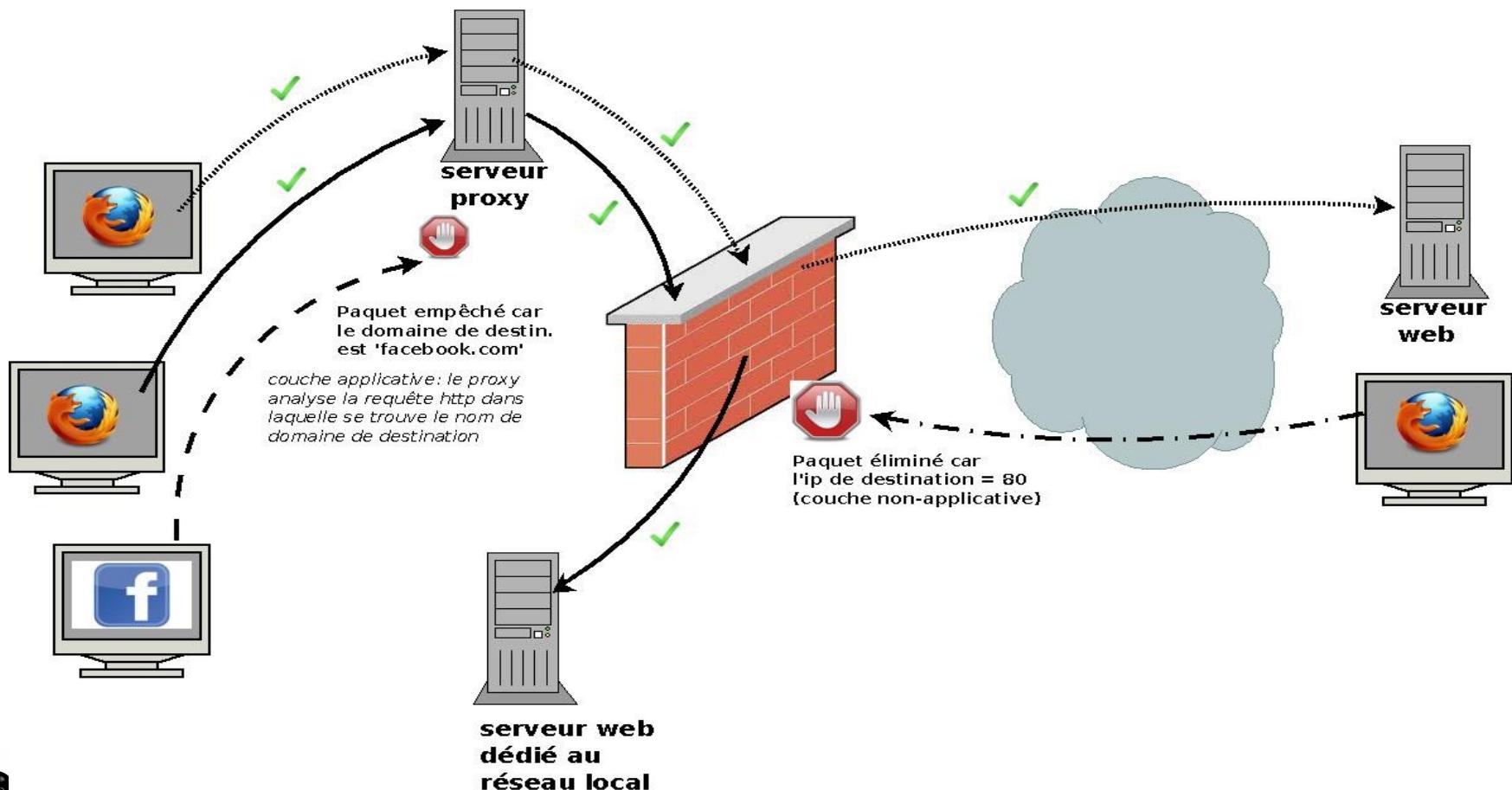
# Pare-feu dit "proxy"

- Il fonctionne au niveau de la couche "applicative".
- Il agit en lieu et place du client.
- La connexion ne peut s'établir que de l'intérieur vers l'extérieur.
- Le proxy se contente d'un genre de port forwarding (ou port mapping).
- Il permet de préserver la bande passante (utilisation d'un cache).
- Il peut demander des authentifications.
- Il peut gérer des listes d'accès (*access lists*).
- Il peut filtrer les réponses sur base de leur contenu.



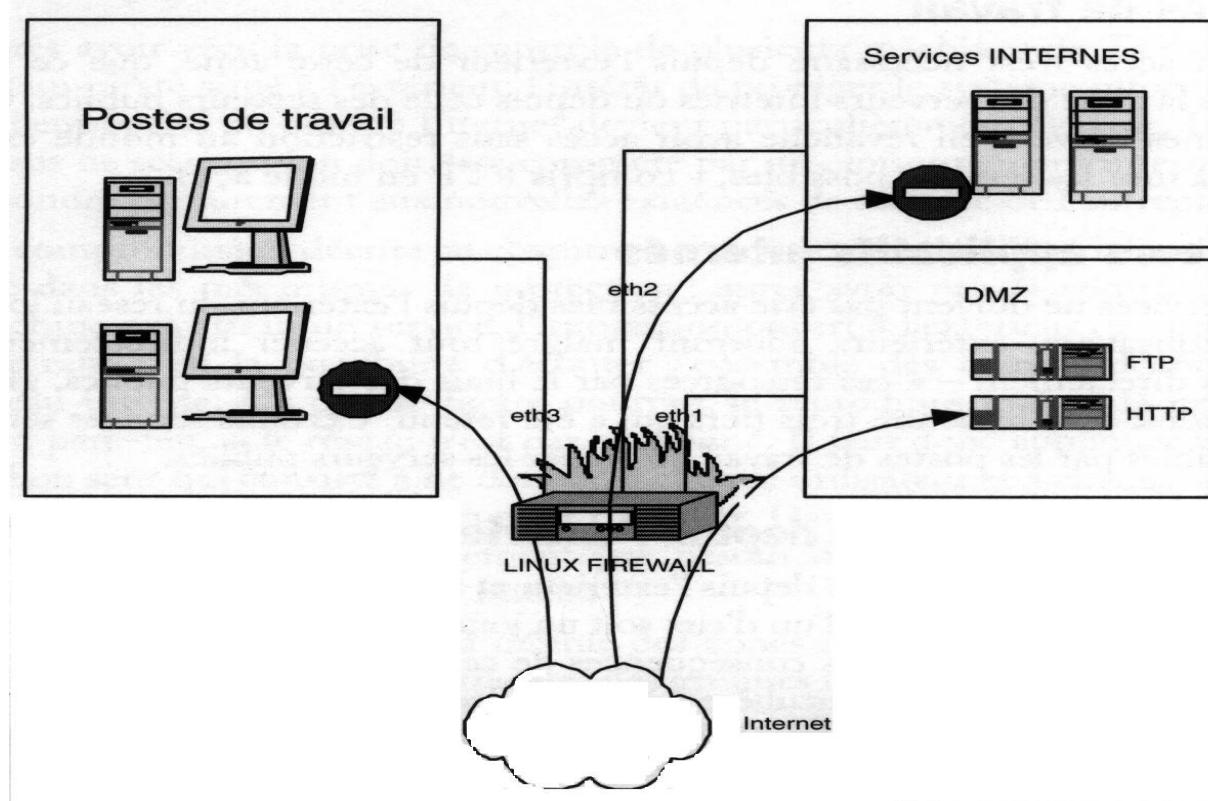
# Pare-feu dit "proxy" (suite)

- Exemple: Soit accepter le port mapping du trafic http sauf celui destiné à certains réseaux sociaux.



# Topologie

## Topologie à un seul pare-feu

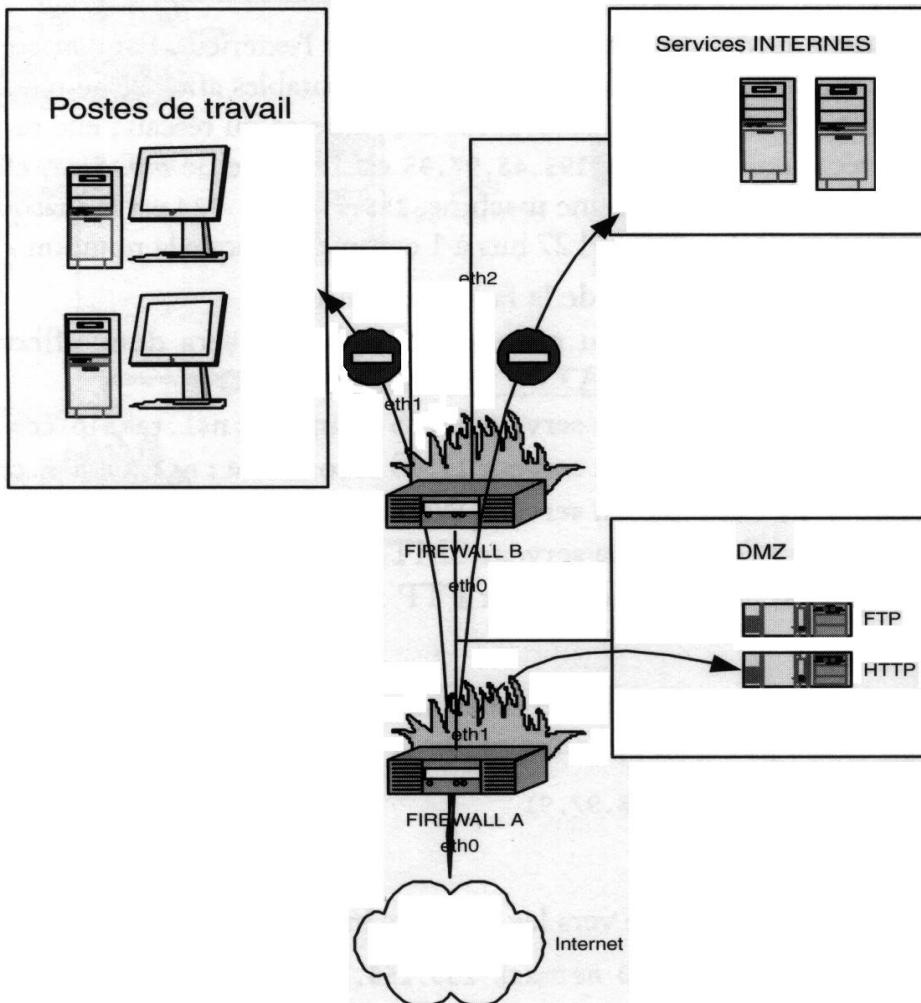


- ☺ Coût réduit (un seul équipement)  
Un seul pare-feu à configurer et à maintenir...
- ☹ Une seule barrière de protection.



# Topologie

## Topologie à double pare-feu



Firewalling (JL Gouwy)

- 😢 Coût plus important.  
Deux pare-feu à configurer et à maintenir.
- 😊 Deux barrières de protection. En cas de compromission des serveurs de la DMZ<sup>(1)</sup>, les postes de travail et les serveurs internes restent protégés<sup>(2)</sup>.

(1) *DMZ (DeMilitary Zone): Réseau sur lequel on installera généralement les serveurs de l'entreprise pouvant être joints depuis l'internet.*

(2) *On mettra souvent en place 2 firewalls de type différents afin d'empêcher que la compromission de l'un permette celle de l'autre (vulnérabilité différentes).*



# Netfilter & IPtables

## Définition

- Pour fournir les fonctionnalités de filtrage disponibles sous Linux à partir du noyau 2.4.
- Netfilter est implémenté au niveau des couches réseaux du noyau Linux. Il effectue le filtrage proprement dit.
- Iptables fournit les commandes nécessaires à la programmation des filtres.

## Fonctionnalités

- Filtrage statique (stateless).
- Filtrage dynamique (statefull) ou suivi de connexion.
- Traduction d'adresse (SNAT et masquerade, DNAT et port forwarding).
- Marquage et manipulation de paquets (plus rarement utilisé).
- Journalisation



# Netfilter & Iptables (suite)

## Filtrage statique

- Utiliser dans un firewall dit "**stateless**" (sans suivi de connexion).
  - Le firewall décide d'accepter ou de refuser un paquet en se basant uniquement sur les informations que ce dernier contient.
- ☺ Pour chaque paquet, il suffit de regarder:
- les Ip source et destination
  - les ports source et destination
  - pour une connexion tcp, l'état des bits, en particulier SYN et ACK
- Peu gourmand en ressources CPU et mémoire
- ☹ Des paquets falsifiés peuvent facilement tromper le firewall  
(pas de suivi de connexion).



# Netfilter & Iptables (suite)

## Filtrage statique (suite)

### Exemple

Soit éliminer tout le trafic non-http de l'intranet vers l'Internet  
(eth0 sur le réseau interne et eth1 sur Internet).

```
iptables -t filter -P FORWARD DROP(1)
```

```
iptables -t filter -A FORWARD -i eth0 -p tcp --dport 80 -j ACCEPT(2)
```

```
iptables -t filter -A FORWARD -i eth1 -p tcp --sport 80 -j ACCEPT(3)
```

<sup>(1)</sup> Stratégie par défaut (*default policy -P*) pour la chaîne FORWARD de la table FILTER  
(il s'agit d'éliminer les paquets par un DROP).

<sup>(2)</sup> Ajout d'une règle (-A) pour autoriser toutes les requêtes HTTP vers l'extérieur.

<sup>(3)</sup> Ajout d'une règle (-A) pour autoriser toutes les réponses HTTP vers l'intérieur  
(le chemin de retour !).



# Netfilter & Iptables (suite)

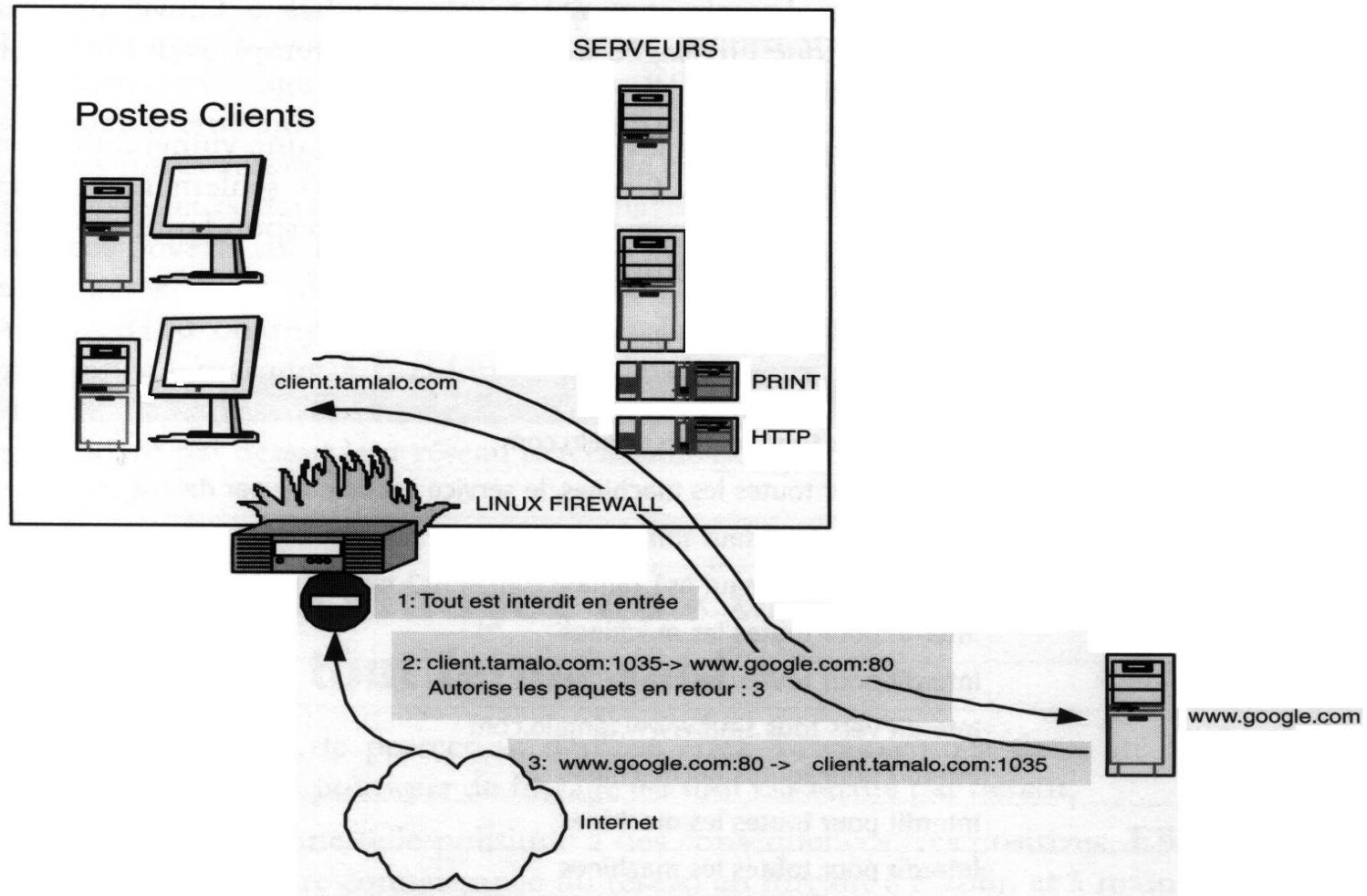
## Filtrage dynamique

- Utiliser dans un firewall dit "**statefull**" (avec suivi de connexion).
- Le firewall garde en mémoire (/proc/net/ip\_conntrack) les échanges de paquets en cours.
  - ☺ Il est donc capable d'autoriser l'entrée de paquets qui sont des réponses à des connexions initiées depuis l'intranet.
    - des paquets falsifiés peuvent difficilement tromper le firewall (suivi de connexion).
    - possibilité de faire du suivi de connexion pour des protocoles complexes où les retours de connexions n'utilisent pas le même couple de ports que la connexion aller (ex. FTP).
  - ☹ Plus gourmand en ressources CPU et mémoire car il doit garder une "pile" des connexions actives.



# Netfilter & Iptables (suite)

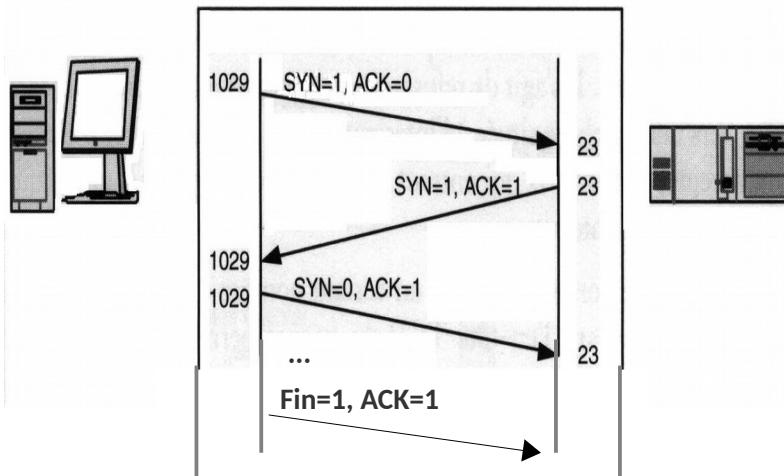
## Filtrage dynamique (suite)



# Netfilter & Iptables (suite)

## Filtrage dynamique (suite)

Ouverture de communication TCP (ex. conversation telnet)



Lorsqu'un paquet TCP contient le flag **SYN**, c'est que c'est une **nouvelle connexion** qui commence.

Lorsqu'un paquet TCP contient les flags **SYN et ACK**, c'est que la **connexion est acceptée**.

Lorsqu'un paquet ne contient que le flag **ACK**, c'est que la **connexion se continue**.

Lorsqu'un paquet contient les flags **FIN et ACK**, c'est que la **connexion se termine**.

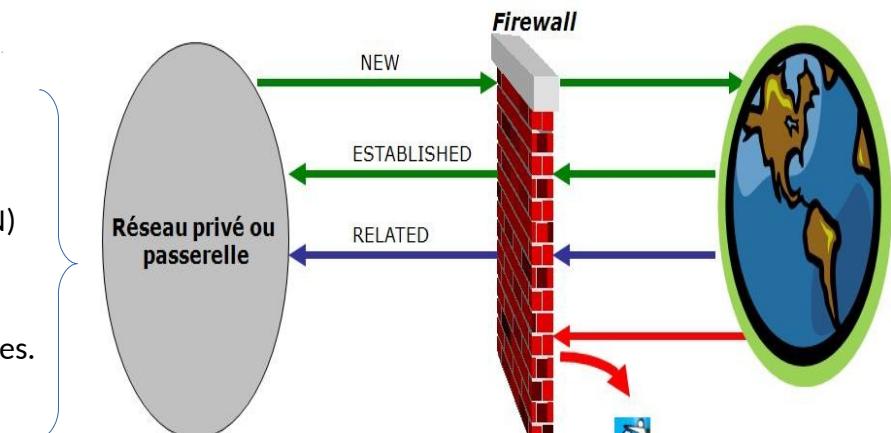
On en déduit les états de connexions

**NEW** Nouvelle connexion. Elle contient le flag SYN.

**ESTABLISHED** Connexion déjà établie (ne contient pas SYN ni FIN)

**RELATED** Connexion en relation avec une autre déjà établie (ex. ftp actif)

**INVALID** Le paquet n'appartient à aucune de ces 3 catégories.



Firewalling (JL Gouwy)



# Netfilter & Iptables (suite)

## Filtrage dynamique (suite)

### Constatations

- Si on n'a pas de serveur, interdire à priori l'entrée de toute connexion NEW. Eventuellement, nous pourrons créer des exceptions pour, par exemple, le port ssh si on dispose d'un serveur ssh ...
- En revanche, toute connexion ESTABLISHED ou RELATED devra pouvoir entrer depuis le Net.
- Dans l'autre sens, du Lan vers le Net, les paquets NEW doivent passer (de même que pour ESTABLISHED et RELATED).
- Les paquets INVALID pourront être journalisés (loggés).



# Netfilter & Iptables (suite)

## Filtrage dynamique (suite)

### Problème avec l'UDP

Impossible de définir l'état d'un échange UDP (car protocole orienté non connexion).

On pourra seulement gérer un "**timer**" pour décider de l'état d'un paquet UDP.

Ex. Une requête DNS depuis le réseau privé

- Le 1<sup>er</sup> paquet à destination du port 53 (DNS) est laissé passer et nous le qualifions de NEW. Il déclenche un timer.
- Si avant expiration du timer, nous recevons un paquet UDP du serveur DNS, nous le qualifions de paquet ESTABLISHED.



# Netfilter & Iptables (suite)

## Filtrage dynamique (suite)

### Exemple

```
iptables -A FORWARD -i eth0 -o eth1 -m state  
          --state NEW,ESTABLISHED,RELATED -j ACCEPT (1)  
          ou  
iptables -A FORWARD -i eth0 -o eth1 -m state --state ! INVALID -j ACCEPT  
  
iptables -A FORWARD -i eth1 -o eth0 -m state  
          --state ESTABLISHED,RELATED -j ACCEPT (2)
```

<sup>(1)</sup> *Toutes les connexions qui sortent du LAN vers le Net sont acceptées.*

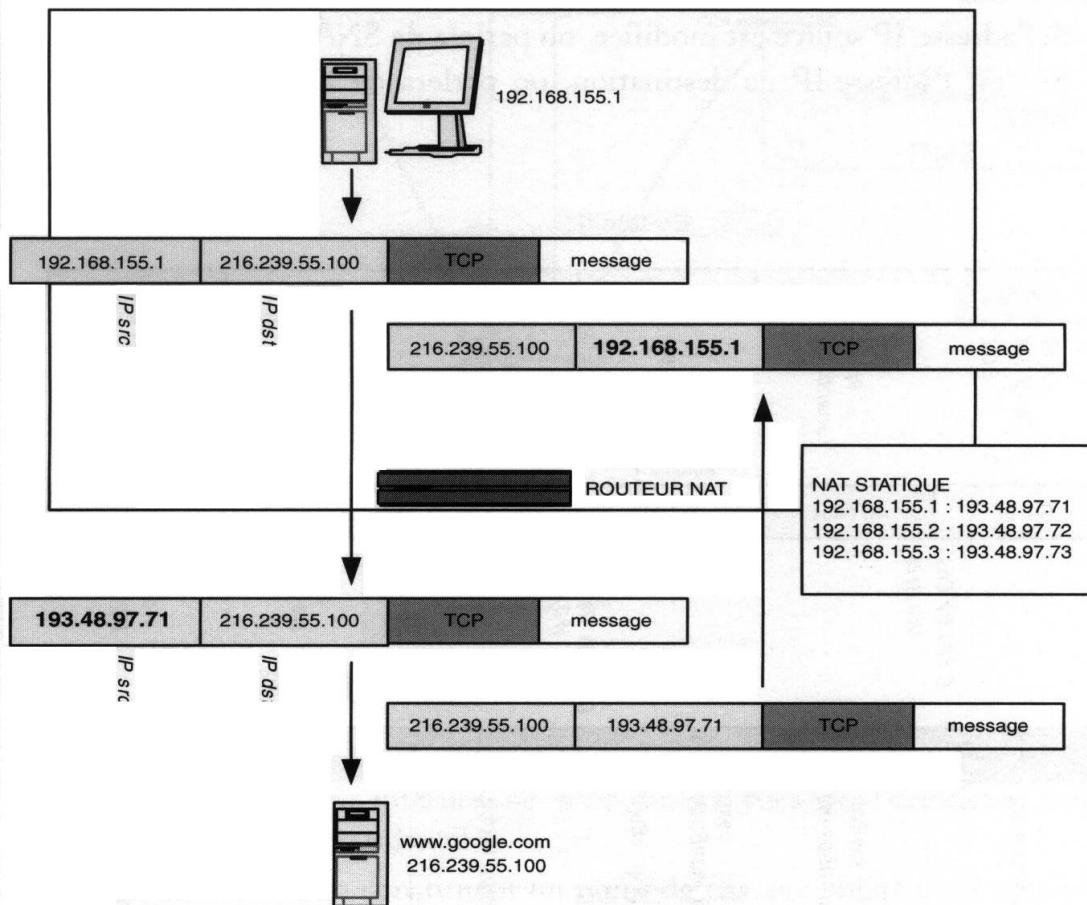
<sup>(2)</sup> *Seules les connexions déjà établies ou en relation avec des connexions établies sont acceptées venant du Net vers le LAN.*



# Netfilter & Iptables (suite)

## Traduction d'adresse

NAT STATIQUE: Ou comment permettre à une machine d'IP privée de pouvoir sortir ?



☺ Simple (peu de ressources CPU et mémoire).

☹ Pour chaque IP interne, il faudrait posséder une IP routable.



# Netfilter & Iptables (suite)

## Traduction d'adresse (suite)

### Exemples (Nat statique)

```
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 193.47.97.70
```

→ Pour remplacer les adresses sources par 193.47.97.70

```
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 193.47.97.70-193.47.97.73
```

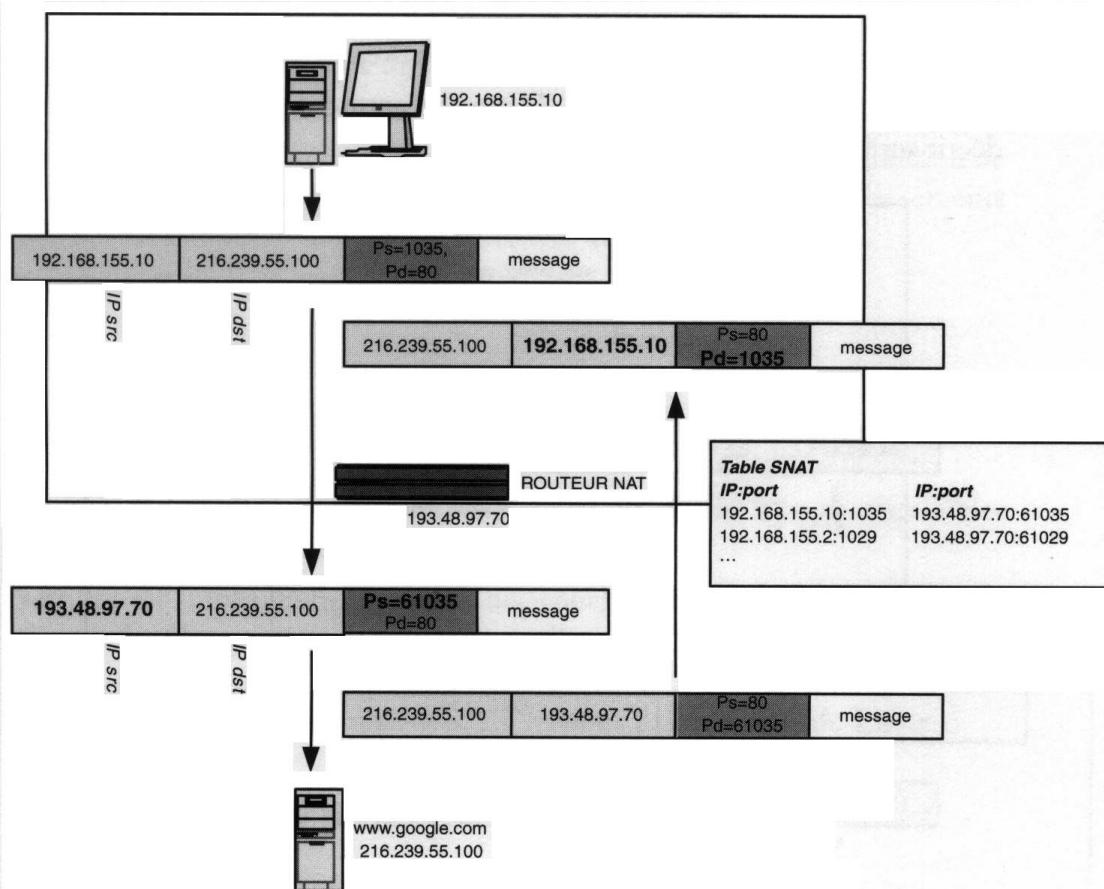
→ Pour remplacer les adresses sources par 193.47.97.70, 193.47.97.71,  
193.47.97.72, 193.47.97.73



# Netfilter & Iptables (suite)

## Traduction d'adresse (suite)

NAT DYNAMIQUE: Ou comment permettre à un réseau d'IP privées de pouvoir sortir ?



Technique appelée:

### "IP masquerading"

☺ Pour chaque IP interne, il ne faut plus qu'une IP routable.

→ Economie d'ip routables.

☺ Une traduction de port (PAT) est nécessaire pour pouvoir restituer le paquet à la bonne machine.

→ Plus complexe (plus de ressources CPU et mémoire).

# Netfilter & Iptables (suite)

## Traduction d'adresse (suite)

Exemple (Nat dynamique: Masquerading)

Soit éliminer tout le trafic non-http de l'intranet vers l'Internet  
(eth0 sur le réseau interne et eth1 sur Internet).

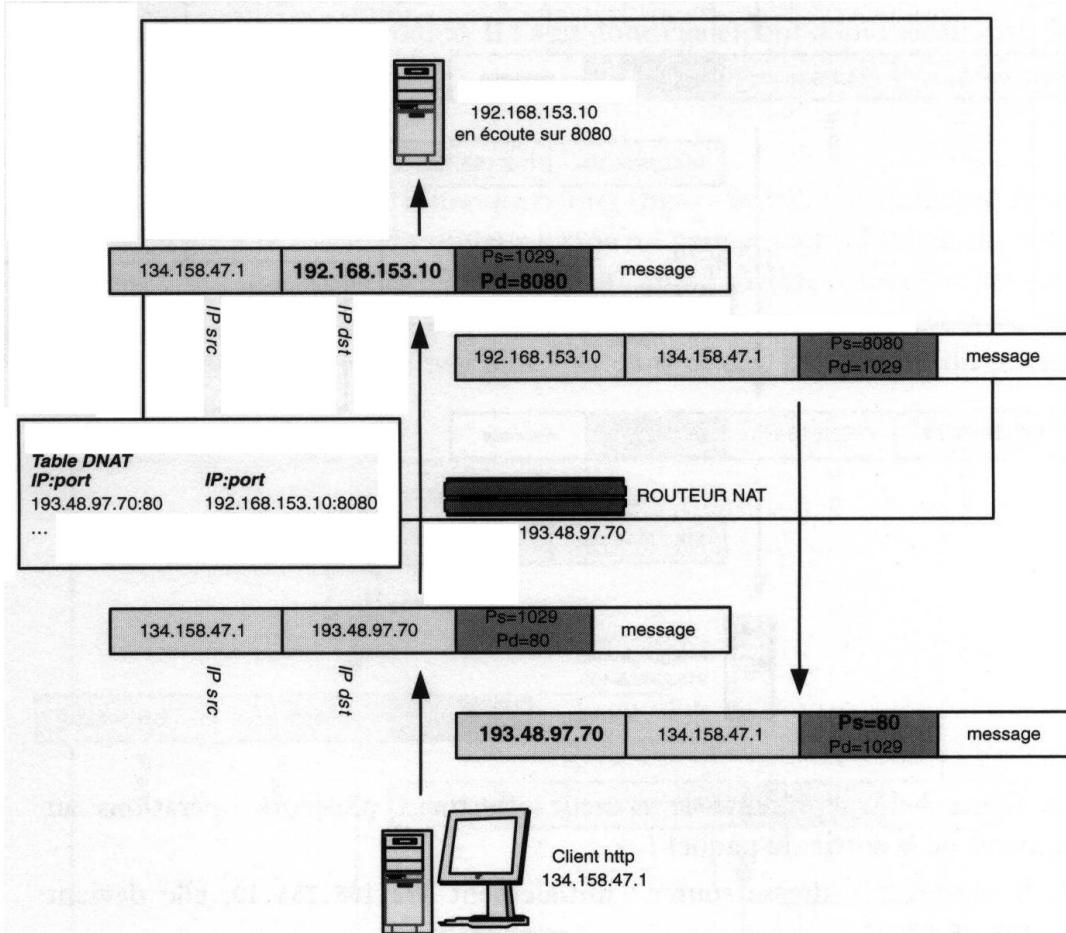
```
iptables -t nat -P POSTROUTING DROP
iptables -t nat -A POSTROUTING -o eth1 -p tcp --dport 80 -j MASQUERADE
```



# Netfilter & Iptables (suite)

## Traduction d'adresse (suite)

Destination NAT: Pour la mise en place d'un serveur "naté"



Technique appelée:  
**"Port forwarding"**

Souvent utilisée pour la mise en place de services démilitarisés.

☺ Le serveur "naté" n'est pas en frontend (càd pas directement accessible de l'extérieur)



# Netfilter & Iptables (suite)

## Traduction d'adresse (suite)

### Exemple (DNAT)

Soit renvoyer tout le trafic http de l'Internet vers un serveur situé sur le réseau interne.

(eth0 sur le réseau interne et eth1 sur Internet).

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT  
        --to-destination 192.168.153.10:8080
```



# Netfilter & Iptables (suite)

## Journalisation

- Utile pour la mise au point des règles de filtrage.
- IPtables permet d'envoyer un message au système syslog quand un paquet coïncide avec une règle de type LOG.
- Cette règle doit être **positionnée avant** toute règle capable de stopper le parcours d'un paquet.
- Lorsqu'une règle de type LOG est vérifiée, **le processus de parcours de règles n'est pas arrêté**.



# Netfilter & Iptables (suite)

## Journalisation (suite)

### Exemple

Pour les paquets concernés par cette règle, un message sera inscrit dans les fichiers de trace du syslog. Ils seront préfixés par le mot clé WEB et auront un degré de priorité INFO.

```
iptables -I -p tcp -s 192.168.155.1 -d 192.168.155.2 --dport 80  
          -j LOG --log-levelinfo info --log-prefix WEB
```



# Les concepts d'Iptables

## Tables, chaînes et règles

- Chacune des fonctionnalités d'IPtables (filtrage, NAT ,marquage) est gérée par une TABLE.
- Chaque table contient un certain nombre de CHAINES.
- Chaque chaîne contient un certain nombre de REGLES.



# Les concepts d'Iptables (suite)

## La table "FILTER"

**Table "filter"** → pour la fonction de filtrage

INPUT      Règle 1

...

Règle n    Exemple

OUTPUT     Règle 1

...

Règle n

FORWARD   Règle 1

...

Règle n

`iptables -t filter -A INPUT -p tcp -s 10.0.0.2 -j DROP`

Nom de la table  
(par défaut c'est  
la table filter).

Ajout de  
cette règle  
à la chaîne  
INPUT.

Sélection du  
paquet  
(matching).

Action.

Règle.

Chaînes  
prédéfinies...

... pouvant  
contenir un certain  
nombre de règles.



# Les concepts d'Iptables (suite)

## La table "NAT"

**Table "nat"** → pour la fonction de traduction d'adresse

PREROUTING      Règle 1

...  
Règle n

### Exemple

POSTROUTING      Règle 1

...  
Règle n

OUTPUT      Règle 1

...  
Règle n

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

Nom de la  
table.

Ajout de  
cette règle à la  
chaîne  
POSTROUTING

Sélection du  
paquet  
(matching).

Action.

Règle.

Chaînes  
prédéfinies...

... pouvant  
contenir un certain  
nombre de règles.



# Les concepts d'Iptables (suite)

## La table "MANGLE"

**Table "mangle"** → pour la fonction de marquage et de manipulation de paquets

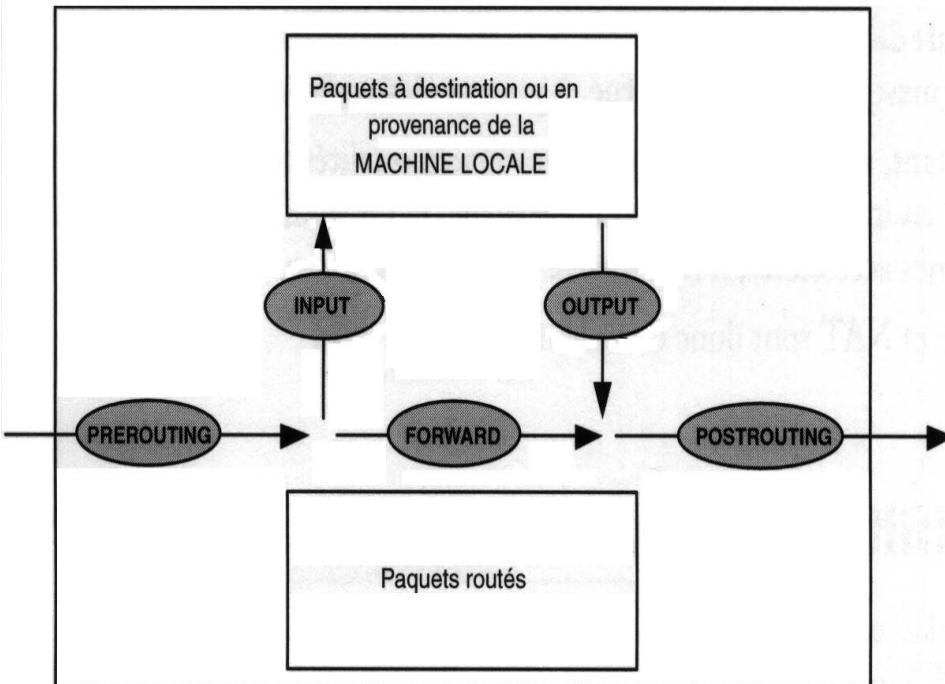
PREROUTING	Règle 1	
	...	
	Règle n	
POSTROUTING	Règle 1	
	...	
	Règle n	
INPUT	Règle 1	Hors cadre du cours
	...	
	Règle n	
OUTPUT	Règle 1	
	...	
	Règle n	
FORWARD	Règle 1	
	...	
	Règle n	



# Les concepts d'Iptables (suite)

## Hook points

- Les chaînes prédéfinies représentent 5 embranchements (hook points) dans la gestion du flux des paquets.
- Les règles définies pour chacune d'elles donnent une opportunité de modifier ou de contrôler le flux de paquets.



Le paquet arrive sur une interface. Les règles appliquées sur **PREROUTING** seront exécutées avant d'analyser l'ip du paquet pour prendre la décision de routage. Cas typique : **DNAT**.

Les règles appliquées sur **POSTROUTING** seront exécutées juste avant d'envoyer le paquet sur l'interface de sortie, alors que le paquet est déjà routé. Cas typiques : **SNAT et MASQUERADE**.

Le paquet n'est pas destiné à la machine locale, mais doit être relayé sur une autre interface. Les règles appliquées sur **FORWARD** ne concernent pas les paquets à destination ou venant de la machine locale.

Les règles appliquées sur **INPUT** concernent tout paquet à destination de la machine locale.

Les règles appliquées sur **OUTPUT** concernent tout paquet en provenance de la machine locale.



# Les concepts d'Iptables (suite)

## Quelques actions (cibles)

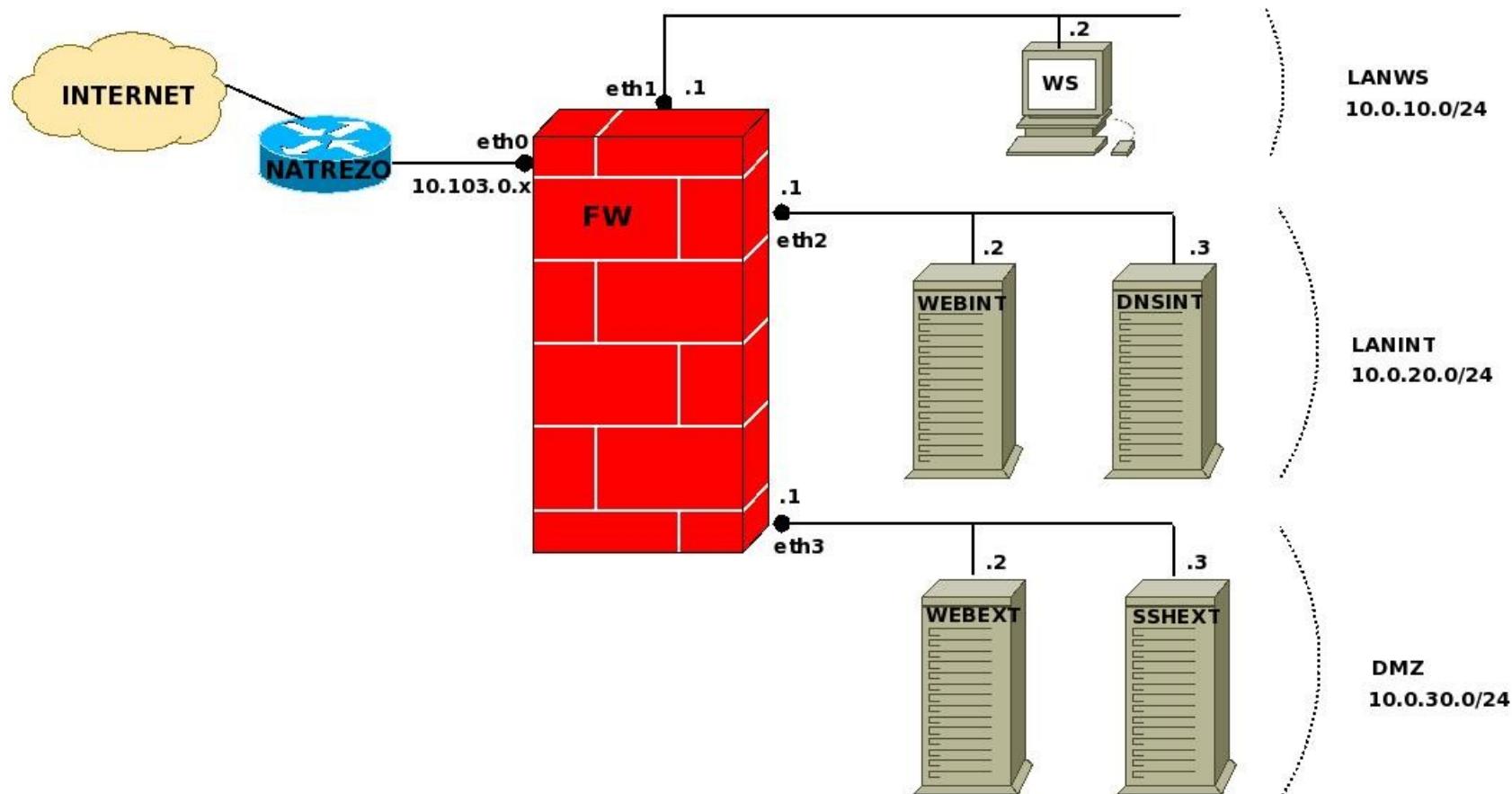
-j ACCEPT	Le paquet est accepté.
-j DROP	Le paquet est rejeté.
-j REJECT	Le paquet est rejeté, l'expéditeur est averti de l'indisponibilité du service.
-j QUEUE	Le paquet est envoyé à une application.
-j LOG	Le paquet est envoyé au système "syslog".
-j SNAT	L'adresse source du paquet est translatée.
-j DNAT	L'adresse destination du paquet est translatée.
-j MASQUERADE	L'adresse & n° port de la source du paquet est translatée.
-j REDIRECT	Le paquet est redirigé d'un port vers un autre port au sein de la machine locale.



# Exercices

## Exercice 1: Firewall stateless

- Préparez la maquette suivante:



# Exercices (suite)

## Exercice 1: Firewall stateless (suite)

- Sur WEBINT et WEBEXT, installez un serveur Apache écoutant sur le port 80 et présentant la page d'accueil configurée par défaut lors d'une requête http sur ceux-ci.
- Sur DNSINT, installez un serveur DNS de cache écoutant sur le port 53 et dont les forwarders sont les deux DNS du FAI.
- Sur SSHEXT, installez un serveur SSH écoutant sur le port 22 et acceptant les connexions ssh par mot de passe (par défaut, il existe déjà...).
- Sur FW, installez un serveur SSH écoutant sur le port 22 et acceptant les connexions ssh par mot de passe (par défaut, il existe déjà...).
- Sur FW, configurez un pare-feu répondant aux spécificités suivantes:
  - Politique du "Tout est fermé": par défaut le pare-feu bloque tout.
  - Les services locaux sur celui-ci doivent pouvoir communiquer entre eux.



# Exercices (suite)

## Exercice 1: Firewall stateless (suite)

- Il autorise toute requête dns venant de n'importe quelle machine d'un des réseaux privés pour être résolue par le serveur de cache (DNSINT).
- L'accès sécurisé (ssh) sur FW est autorisé uniquement à partir des machines du réseau LANINT.
- Le serveur web de WEBINT ne sera accessible que par les machines du réseau LANWS.
- Les services spécifiques de la DMZ ne seront accessibles que de l'extérieur.



# Exercices (suite)

## Exercice 1: Firewall stateless (suite)

### REMARQUES

- Lors du démarrage du pare-feu, on effacera d'abord toutes les règles des chaînes de toutes les tables :

Options:

- F : pour les chaînes prédéfinies.
- X : pour les chaînes 'utilisateurs' (pas étudiées ici).

- Ecrivez toutes vos règles au sein d'un script bash /etc/init.d/fwless et répondant aux appels suivants :

```
# /etc/init.d/fwless start    → démarrage du pare-feu
# /etc/init.d/fwless stop     → démarrage du pare-feu "tout ouvert"
# /etc/init.d/fwless status   → affichage des règles en Ram
                                         (Choisir une des options de debug
                                         -L, -L -v, -S)
```



# Exercices (suite)

## Exercice 2: Firewall statefull

- Idem 'Exercice 1' mais en rendant le pare-feu le plus 'statefull' possible.
- Ecrivez toutes vos règles au sein d'un script bash /etc/init.d/fwfull et répondant aux appels suivants :

# /etc/init.d/fwless start	→ <i>démarrage du pare-feu</i>
# /etc/init.d/fwless stop	→ <i>démarrage du pare-feu "tout ouvert"</i>
# /etc/init.d/fwless status	→ <i>affichage des règles en Ram</i> <i>(Choisir une des options de debug</i> -L, -L -v, -S)



# Références

## WEBOGRAPHIE

<http://irp.nain-t.net/doku.php/130netfilter:start>

## BIBLIOGRAPHIE

Administration réseau sous Linux (O'reilly - T.Bautts, T. Dawson, G.N. Purdy)

Sécuriser un réseau sous Linux (Eyrolles 2<sup>ème</sup> éd. - B.Boutherin, B. Delaunay)

Sécurité Firewalls (Formation TechnofuturTic - J. Van Rysselberghe)

