

Table des matières

I. Introduction.....	2
II. Fonctionnement.....	2
1. Principe général.....	2
2. Mode routed.....	2
a) Principe.....	2
b) Limitations dues à Windows.....	4
c) Donner accès au réseau du serveur ou des clients.....	5
3. Mode bridged.....	5
a) Ethernet Bridging.....	5
b) Principe.....	6
c) Fonctionnalités.....	7
d) Limitation de Windows.....	7
4. Analogie avec une liaison physique.....	7
5. Préauthenticatio.....	8
6. Avantages du Bridging.....	8
a) Inconvénients du Bridging.....	8
b) Avantages du Routing.....	8
c) Inconvénients du Routing.....	9
III. Mise en place.....	9
1. Configuration du noyau.....	9
2. Installation.....	9
3. Test.....	10
4. Génération des clés.....	10
5. Et si je met une passphrase pour crypter mes clés.....	12
6. Configuration du serveur en mode Routed.....	12
7. Configuration du serveur en mode Bridged.....	13
a) Script de démarrage supplémentaires.....	13
b) Configuration du serveur.....	15
8. Configuration du client.....	16
a) Configuration du client sous Linux.....	16
b) Configuration du client sous Windows.....	16
c) Configuration en mode Routed.....	16
d) Configuration en mode Bridged.....	17
9. Utilisation.....	18
10. Pare-feu serveur.....	18
a) Mode routed.....	18
i. Connexion sur le serveur.....	18
ii. Connexion au réseau interne derrière le serveur.....	19
b) Mode bridged.....	19
i. Si vous utilisez un noyau 2.4.....	19
ii. Règles de filtrage.....	19
c) Pare-feu client.....	20
d) Règles à éviter.....	21
11. Considération de sécurité.....	21

IV. Annexe : le protocole SSL et OpenSSL.....	21
1. Qu'est-ce qu'OpenSSL ?.....	21
2. Connexion SSL.....	21
V. Bibliographie.....	24

I. Introduction

OpenVPN est une solution qui se base sur SSL. Cela permet d'assurer deux choses à la fois, sans avoir besoin de beaucoup de logiciel côté client :

- l'authentification du client et du serveur
- la sécurisation du canal de transmission

Il permet par exemple de résoudre les problèmes de NAT en offrant la même protection qu'IPSec mais sans les contraintes.

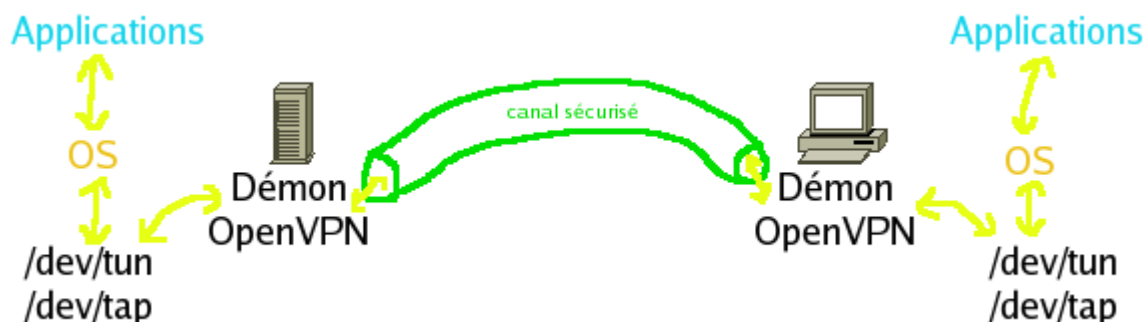
II. Fonctionnement

Il existe deux modes de fonctionnement de OpenVPN :

- le mode routé (Routed) qui permet de connecter des utilisateurs itinérants à un réseau interne
- le mode ponté (Bridged) qui permet de relier entre eux deux sous-réseaux

1. Principe général

Le principe général est de relier les /dev/tun ou /dev/tap du serveur et des clients par un canal sécurisé par OpenSSL.



2. Mode routed

a) Principe

Le principe général (sans tenir compte des sous réseaux /30) d'OpenVPN est le suivant :

- le serveur crée une connexion Point à Point entre lui et le système d'exploitation, par exemple x.y.z.1 pour l'OS et x.y.z.2 pour le serveur OpenVPN (passerelle vers le réseau VPN). On voit donc sur le serveur
 - une interface tun0 qui a pour IP x.y.z.1 et qui est liée Point à Point avec une IP x.y.z.2 (le serveur VPN). Le serveur VPN récupère donc les paquets arrivant sur cette interface et

envoi des paquets depuis x.y.z.2 vers x.y.z.1. x.y.z.1 est donc pingable dans la mesure où c'est une interface gérée par l'OS du serveur. `ifconfig tun0` donne

```
Lien encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet adr:x.y.z.1 P-t-P:x.y.z.2 Masque:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
```

- une route pour que l'OS sache que x.y.z.2 (le serveur VPN) est l'autre bout du Point à Point sur tun0 et une route pour indiquer que les paquets à destination du réseau VPN (n'importe quel client VPN) doivent passer par tun0 et par le serveur VPN (qui sert de passerelle entre les machines clients et la machine serveur). `route` donne :

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
x.y.z.2	*	255.255.255.255	UH	0	0	0	tun0
x.y.z.0	x.y.z.2	255.255.255.0	UG	0	0	0	tun0

- pour chaque client, le serveur crée une connexion Point à Point avec le client, par exemple x.y.z.6 pour le client et x.y.z.5 pour le serveur VPN (je reviendrais sur l'attribution du numéro d'hôte dans la section suivante). On voit sur le client :

- une interface tun0 qui a pour IP x.y.z.6 et qui est liée Point à Point avec une IP x.y.z.5 (le serveur VPN). Le client VPN récupère donc les paquets arrivant sur cette interface et envoie des paquets depuis x.y.z.6 vers x.y.z.5. x.y.z.6 est donc pingable dans la mesure où c'est une interface gérée par l'OS du client. `ifconfig tun0` donne

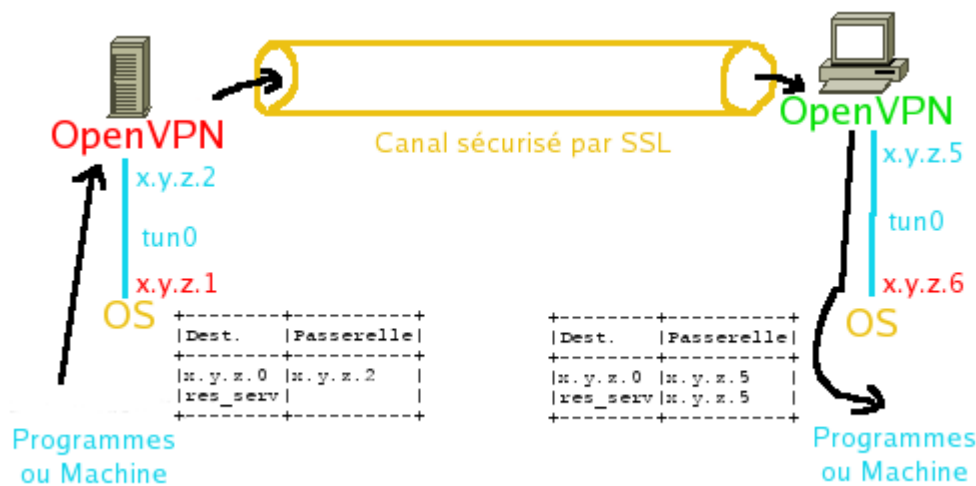
```
Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet adr:x.y.z.6 P-t-P:x.x.z.5 Masque:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
```

- une route pour que l'OS sache que x.y.z.5 (le serveur VPN) est l'autre bout du Point à Point sur tun0, une route pour indiquer que les paquets à destination du réseau VPN (principalement le serveur VPN mais aussi entre client) doivent passer par tun0 et par le serveur VPN (qui sert de passerelle entre les machines clients et la machine serveur) et autant de route que l'on veut pour indiquer que les paquets à destination d'un réseau se trouvant derrière la machine serveur doivent passer par le réseau VPN via tun0 et donc via le serveur VPN. `route` donne :

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
172.20.9.5	*	255.255.255.255	UH	0	0	0	tun0
172.20.9.0	172.20.9.5	255.255.255.0	UG	0	0	0	tun0
192.168.0.0	172.20.9.5	255.255.255.0	UG	0	0	0	tun0

Le serveur VPN a donc deux IP virtuelles (non pingables) ce qui lui permet de servir de passerelle entre la machine cliente qui a une IP réelle sur une interface virtuelle et la machine serveur qui a aussi une IP réelle sur une interface virtuelle (la machine, pas seulement le serveur OpenVPN).

Cela peut se résumer par le schéma suivant :



b) Limitations dues à Windows

OpenVPN est obligé d'allouer un sous réseau /30 pour chaque client qui se connecte en mode routed pour garder la compatibilité sur tous les OS. Ceci est dû aux limitation du mode d'émulation TUN du pilote TAP-Win32 sous Windows.

On peut néanmoins indiquer à OpenVPN de ne pas avoir ce comportement si l'on sait qu'il n'y a pas de clients Windows. On indique alors dans le fichier de configuration **ifconfig-pool-linear**.

Dans la version 2.0, le serveur peut accueillir plusieurs clients sur la même interface TUN. Cela se réalise de la façon suivante :

- sur le serveur, il y a un lien point à point entre le serveur VPN et le système d'exploitation. C'est le tun0 que l'on voit en faisant un ifconfig sur le serveur VPN. Cette liaison récupère le premier sous réseau /30 à savoir les adresses x.y.z.0, x.y.z.1, x.y.z.2 et x.y.z.3. L'adresse IP du serveur dans ce sous réseau est une IP réelle que l'OS gère comme sur une interface eth, elle peut donc être pinguée.
- puis il y a une liaison point à point vers chaque client qui commence donc à l'adresse d'hôte x.y.z.4 et chaque client prend quatre adresses d'hôte. C'est le tun0 que l'on voit quand on fait un ifconfig sur le client. Le premier sous réseau client est donc x.y.z.4/30 (x.y.z.4, x.y.z.5, x.y.z.6 et x.y.z.7). L'IP du serveur dans ces sous réseau est une adresse virtuelle qui ne sert à OpenVPN que pour récupérer les paquets venant du client. Elle n'existe donc pas vraiment et ne peut donc pas être pinguée contrairement à l'IP du premier sous réseau entre le serveur et l'OS.

Si tous les OS pouvaient utiliser une vrai liaison point à point sur l'interface TUN, il suffirait d'avoir une adresse IP pour le serveur et une individuelle pour chaque client. Mais sous Windows, on ne peut pas. Cela génère une sorte de gâchie d'IP mais cela permet d'être portable puisque l'on permet 3 adresses sur 4. Il faut donc simuler une liaison point à point par un sous réseau de quatre IP :

- l'adresse du réseau (pas vraiment utile)
- l'adresse du serveur (un bout du point à point)
- l'adresse du client (l'autre bout du point à point)
- l'adresse de diffusion du réseau (encore moins utile)

Le pilote TAP-Win32 inclu un serveur DHCP virtuel qui assigne au client la seconde adresse IP du sous réseau /30 du client. On voit donc la première adresse IP comme ce serveur DHCP virtuel.

c) Donner accès au réseau du serveur ou des clients

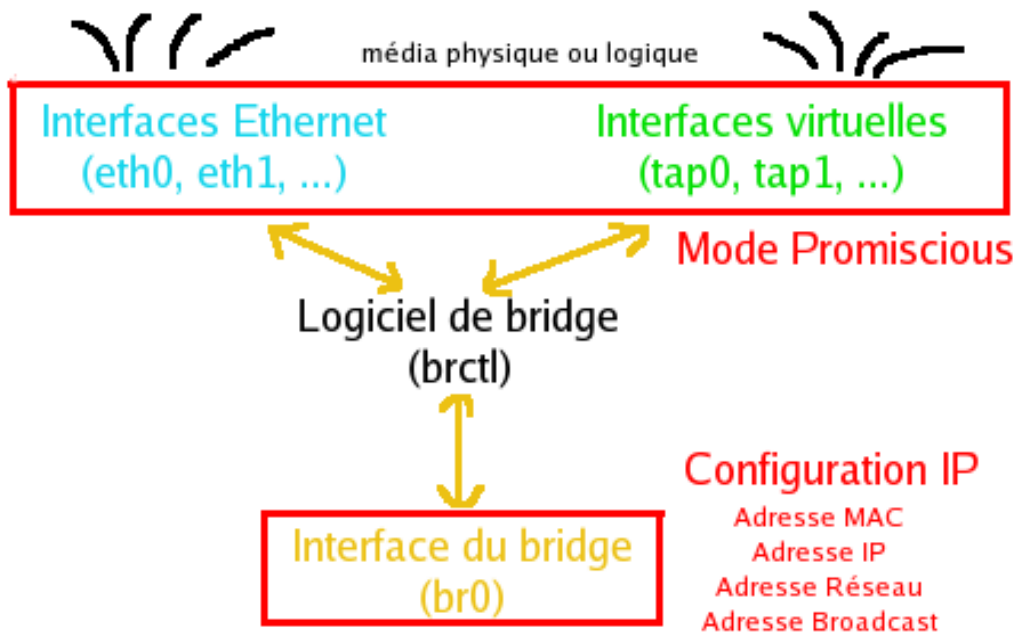
Pour autoriser les clients à accéder au réseau qui se trouve derrière le serveur, il faut indiquer au client que la passerelle pour accéder à ce réseau est l'IP du serveur dans le réseau /30 du client, donc l'adresse IP juste avant celle du client. Cela se fait par la directive push route dans le fichier de configuration.

3. Mode bridged

a) Ethernet Bridging

Ethernet Bridging désigne l'action de regrouper plusieurs interfaces physiques (cartes Ethernet) et/ou virtuelles (TAP) sur un même pont (bridge, une interface de remplacement/regroupement) gérée de manière logicielle afin de partager un même sous réseau sur plusieurs interfaces. C'est en quelque sorte, un switch logiciel qui écoute sur toutes les interfaces pontées et fait le tri, comme un switch pour dispatcher vers la bonne sortie (interface).

On pourrait représenter un pontage comme un aiguillage automatique (un switch) :



La configuration IP se fait alors sur l'interface Bridge et pas sur les interfaces physiques ou virtuelles bridgées. En effet, le bridge a pour effet d'effacer la configuration IP des interfaces qu'elle bridge pour les passer en Promiscuous et donc faire un traitement logiciel des paquets entrants et sortant des interfaces bridgées.

Il faut donc toujours configurer l'interface Bridge avant d'ajouter des interfaces physiques ou virtuelles au bridge sans quoi la configuration IP est perdue.

b) Principe

Le principe du mode bridged est le suivant, par exemple si la carte réseau à bridger est eth0 :

- sur le serveur : on fait un pont br0 (bridge) regroupant l'interface physique eth0 et l'interface virtuelle tap0 pour partager le sous réseau de eth0 avec tap0. Cela permet d'intégrer les clients dans le sous réseau de eth0.
 - on lance le bridge ce qui a pour effet de créer les interfaces br0, tap0 et de ponter tap0 et eth0 sur br0 :
 - br0 obtient la configuration IP de eth0
 - tap0 et eth0 perdent leurs configurations IP/Ethernet (adresse MAC comprise) et passent en mode Promiscuous (remonte tous les paquets arrivant sans faire le tri dans la carte réseau) sans adresse IP

○ on obtient donc les interfaces suivantes par ifconfig :

- eth0 :

```
Lien encap:Ethernet HWaddr 00:50:DA:11:6B:CE
UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
```

- tap0 :

```
Lien encap:Ethernet HWaddr 00:FF:AF:FE:3C:D1
UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
```

- br0 :

```
Lien encap:Ethernet HWaddr 00:50:DA:11:6B:CE
inet adr:192.168.0.25 Bcast:192.168.0.255 Masque:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

○ cela met en place une route indiquant que le réseau interne initialement sur eth0 est maintenant sur br0. Par route :

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
192.168.0.0	*	255.255.255.0	U	0	0	0	br0

- sur le client, OpenVPN crée une interface tap0 reflétant la configuration IP obtenue sur le réseau du eth0 de la machine serveur. Par exemple, si le pool d'adresses Ips pour les clients VPN est entre 192.168.0.129 et 192.168.0.254 (autrement dit le sous réseau 192.168.0.128/25) :

○ on obtient donc l'interface suivante par ifconfig tap0 :

```
Link encap:Ethernet HWaddr 16:31:8C:46:64:35
inet adr:192.168.0.129 Bcast:192.168.0.255 Masque:255.255.255.0
adr inet6: fe80::1431:8cff:fe46:6435/64 Scope:Lien
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

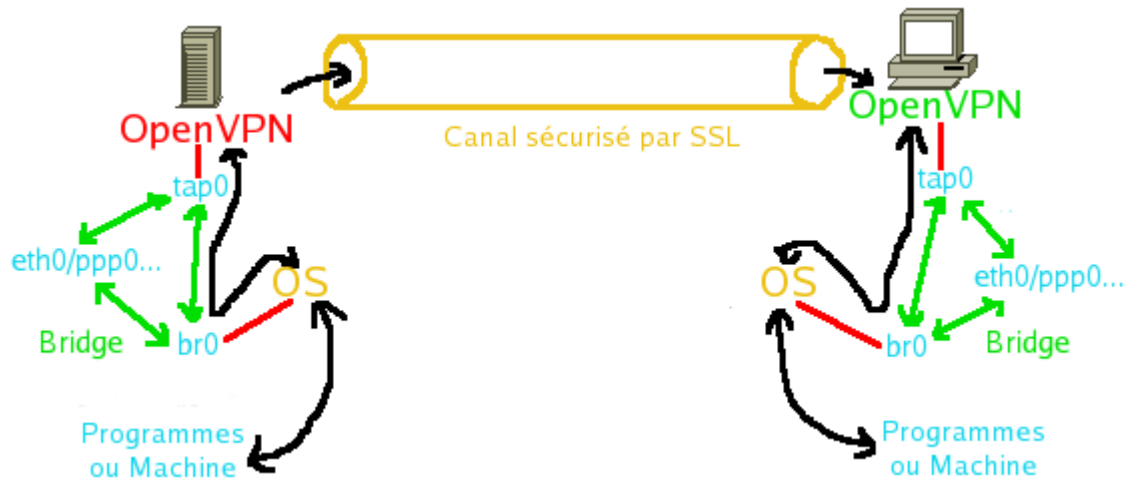
○ cela met en place une route indiquant que le réseau VPN (réseau interne de la machine serveur) est atteint par tap0. Par route :

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
192.168.0.0	*	255.255.255.0	U	0	0	0	tap0

- le serveur OpenVPN sert alors de simple connexion sécurisée entre le tap0 du client et le tap0 du serveur. Tout ce qui est écrit sur l'un est retransmis sur l'autre et vice-versa. En plus, il alloue les adresses IPs des clients, cette adresse étant choisie parmi un pool d'adresses du sous réseau initialement de eth0

Note : dans le cas du bridge, toutes les adresses IP sont réelles et peuvent donc être pinguées.

Cela peut donc se résumer par le schéma suivant :



c) Fonctionnalités

Le mode bridged permet d'obtenir les fonctionnalités d'Ethernet (niveau 2 de la pile IP) comme le transport natif des protocoles de niveau 3 comme NetBIOS ou IPX et pas seulement IP comme dans le mode routed. Cela permet aussi de faire des broadcast sur les deux réseaux reliés. On configure (avec un utilitaire comme brctl sous Linux) le bridge sur la machine client et serveur ce qui permet de relier les deux réseaux qui sont derrière ces deux machines.

Le mode routed est plus simple dans le cas où l'on n'a pas besoin de ces fonctionnalités d'Ethernet.

On peut connecter des machines dans les deux modes pour créer des architectures relativement complexes mais cela n'est pas l'objet de ce guide.

d) Limitation de Windows

Toutes les versions de Windows peuvent servir pour un client. Pour faire un serveur OpenVPN avec Windows, il faut dans ce cas avoir XP, 2003 ou supérieur.

4. Analogie avec une liaison physique

Imaginons que l'on ait un câble direct entre deux ordinateurs A et B situés dans différents lieux. Sur chaque ordinateur, on aura dans la tradition UNIX, `/dev/cableAB` qui devrait être un périphérique réseau. Il suffirait alors de déclarer une route passant par ce périphérique et on pourrait s'en servir comme une interface réseau classique. Le câble relie les deux périphériques sur les deux ordinateurs.

Eh bien c'est ce que TUN et TAP font. C'est exactement comme notre hypothétique `/dev/cableAB`. Et le câble est le démon OpenVPN client et serveur. C'est lui qui sert de liaison entre les deux `/dev/tun`. De plus, dans le cas de OpenVPN, le moyen de liaison entre l'ordinateur A et B peut être ce que l'on veut y compris Internet.

Et OpenVPN relie les deux `/dev/tun` en identifiant d'abord les deux ordinateurs puis en cryptant les données dans des paquets UDP entre ces deux entités. De plus, il récupère le trafic entrant dans `/dev/tun` sur A pour le faire sortir sur B et inversement. Enfin, cela est d'autant plus simple que les

pilotes TUN et TAP ont été prévus pour être accessibles par le niveau 3 (userland) autrement dit par des applications traditionnelles et PAS dans des modules niveau 0 (noyau). Cela permet donc d'être portable contrairement à IPSec qui nécessite de créer des modules noyaux donc différents entre les versions de Linux et pour Windows, je n'en parle même pas.

La différence entre un périphérique TUN et TAP est simple. TUN est un périphérique qui crée une liaison virtuelle IP point à point qui peut donc gérer seulement 2 IP. TAP est un périphérique Ethernet virtuel qui peut donc gérer un réseau virtuel. Si l'on reprend notre /dev/cableAB, un TUN revient à voir une liaison T1 par exemple connectée entre le PC A et le PC B. Un TAP revient alors à avoir un réseau Ethernet reliant les deux ordinateurs A et B.

5. Préauthentification

OpenVPN utilise OpenSSL pour créer le canal crypté.

Le fichier ta.key (tls-auth) permet d'éviter les attaques par Deny of Service. Cela peut se produire lorsque le serveur reçoit énormément de demande d'authentification (qui demande de la mémoire et du temps processeur). En effet, lorsqu'il est configuré pour utiliser cette clé, le client qui n'est encore authentifié par le handshake de TLS (avec les certificats clients et serveur) doit signer ses paquets avec la clé ta.key sans quoi le serveur le supprime sans chercher à aller plus loin dans l'authentification. C'est donc une clé qui est partagée entre les clients et le serveur. Un attaquant qui, par définition, n'a pas la clé ta.key ne peut pas commencer son authentification sur le serveur et donc ne pourra pas lui faire consommer le temps CPU nécessaire à un faux début de handshake TLS. Cela est donc un contrôle avant l'authentification pour ne pas chercher à authentifier une machine dont on est sûr qu'elle ne réussira pas l'authentification.

6. Avantages du Bridging

- Les paquets de diffusions traversent le VPN. Cela permet à des applications comme le partage de fichier par NetBIOS Windows ou l'affichage du voisinage réseau de fonctionner.
- Pas de route à configurer.
- Fonctionne avec tous les protocoles qui se trouvent au dessus d'Ethernet, comme IPv4, IPv6, Netware IPX, AppleTalk, etc.
- Simple à configurer pour les utilisateurs itinérants

a) Inconvénients du Bridging

- Moins rapide que le mode Routing et moins adapté aux réseaux VPN de grande taille

b) Avantages du Routing

- Rapide même sur des réseaux de grande taille
- Permet de régler plus finement le MTU pour un meilleur rendement

c) Inconvénients du Routing

- Les clients doivent utiliser un serveur WINS (comme samba) pour faire marcher le parcours du voisinage réseau entre les réseaux reliés par le VPN
- Les routes doivent être configurées pour chaque sous réseau à atteindre
- Les logiciels qui dépendent du broadcast IP ne verront pas les machines qui sont de l'autre côté du VPN

- Fonctionne uniquement avec Ipv4 et Ipv6 si le pilote TUN/TAP de la plate-forme gère ce dernier.

III. Mise en place

1. Configuration du noyau

Il peut être utile de mettre en place le patch GRSecurity pour votre noyau.

OpenVPN utilise le périphérique TAP/TUN comme interface.

Il faut donc l'avoir activé dans le noyau :

Network device support -> Universal TUN/TAP device driver support

Pour vérifier si ce périphérique est inclus dans le noyau ou dans un module de celui-ci :

```
cat /boot/config-`uname-r` |grep TUN
```

Doit renvoyer au moins une ligne telle que : CONFIG_TUN=y ou CONFIG_TUN=m

Normalement, il devrait y avoir un device /dev/net/tun, sinon, il faut le créer par :

```
[root]# mkdir /dev/net/ && mknod /dev/net/tun c 10 200
```

2. Installation

OpenVPN dépend de :

- OpenSSL et ses composants Dev
- LZO
- brctl si vous voulez avoir le mode ponté

L'installation peut se faire comme d'habitude :

- par compilation (nécessite en plus automake, autotools et autoconf) : `./configure && make && make install`
- en récupérant un paquet RPM, DEB ou emerge par votre yum et autre apt-get

Dans le premier cas, tous les dossiers mentionnés sont dans le répertoire des sources de OpenVPN. Dans le second cas, elles sont dans le dossier /usr/share/doc/openvpn-2.0/.

Il sera alors possible de lancer openvpn par `/etc/init.d/openvpn start`.

Vous pouvez utiliser le script de démarrage que vous trouverez dans le dossier `sample-config-files`.

Pour un démarrage automatique d'OpenVPN à chaque redémarrage taper :

```
chmod 755 /etc/init.d/openvpn
ln -s /etc/init.d/openvpn /etc/rc3.d/S90openvpn
ln -s /etc/init.d/openvpn /etc/rc6.d/K90openvpn
```

3. Test

Une fois l'installation terminée, rendez vous dans le dossier d'installation d'OpenVPN et taper :

```
./openvpn --genkey --secret key
./openvpn --test-crypto --secret key

./openvpn --config sample-config-files/loopback-server
```

Dans une autre console taper :

```
./openvpn --config sample-config-files/loopback-client
```

Si tout ce passe bien, la connexion devrait se terminer au bout d'une ou deux minutes sans messages d'erreurs.

4. Génération des clés

OpenVPN nécessite un certain nombre de clés :

- une clé de pour le protocole d'échange diffie-hellman (*dh1024.key*)
- une clé et un certificat pour l'autorité de certification (le moins cher est de créer la sienne) (*ca.crt, ca.key*)
- une clé et un certificat pour le serveur (*nom_serveur.key, nom_serveur.crt*)
- une clé et un certificat pour chaque client (*nom_client.crt, nom_client.key*)
- une clé qui sert à autoriser les accès au démon (*ta.key*)

Note : les fichiers .key (ou .pem contenant une clé privée) doivent absolument avoir au maximum les droits 600 et appartenir à l'utilisateur qui s'en sert.

Heureusement, OpenVPN fournit un script pour faire cela de façon beaucoup plus simple. Pour cela, allez dans le dossier *easy-rsa* puis :

- éditer le fichier *vars* afin de modifier les informations qui vont être incluses dans les certificats générés (et pas dans *openssl.cnf*), par exemple :

```
export D=/usr/share/doc/openvpn-2.0/easy-rsa
export KEY_CONFIG=$D/openssl.cnf
export KEY_DIR=$D/keys
export KEY_SIZE=1024
export KEY_COUNTRY=FR
export KEY_PROVINCE=Département
export KEY_CITY=Ville
export KEY_ORG="Nom serveur VPN"
export KEY_EMAIL="email_administrateur"
```

- préparer la génération des clés (définition du numéro de série)

```
source ./vars
mkdir keys
```

```
touch keys/index.txt
echo 01 > keys/serial
```

- décompresser le fichier `openssl.cnf` (à décompresser `gzip -d openssl.cnf.gz`) si ce n'est pas déjà fait. Il n'y a rien à modifier dedans contrairement à la génération avec OpenSSL.
- générer les clés et les certificats :

Il faut utiliser des **Common Name** (CN) unique pour chacun des clients et du serveur sinon votre serveur ne fonctionnera pas ! Il faut aussi un **Organization Name** (ON) commun au serveur et aux clients (c'est le nom de votre VPN). Pour les autres renseignements, la valeur par défaut est indiquée entre [] et correspond à ce que vous avez mis dans le fichier `vars`. Lorsque l'on vous demande si les certificats doivent être signés répondez y.

- générer la clé Diffie-Hellman :

```
./build-dh
```

- générer le certificat de l'autorité de certification :

```
./build-ca
```

- générer la clé du serveur :

```
./build-key-server nom_serveur
```

- générer des clés pour les clients :

```
./build-key nom_unique_client
```

Il est important que les fichiers `.key` contenant les clés privées des clients soient transmises par un canal sûr comme SSH. En effet, toute la sécurité de OpenVPN repose sur la bonne transmission de ces fichiers clés. Les fichiers `.crt` ne nécessitent pas une vigilance particulière car ils ne contiennent que des informations publiques.

- générer la clé permettant d'éviter les attaques man-in-the-middle et autorisant l'accès au démon OpenVPN :

```
../openvpn --genkey --secret keys/ta.key
```

- mettre toutes les clés dans le dossier `/etc/openvpn/keys` et les donner à `nobody:nobody` (ou `nobody:nogroup` sous Debian)

```
mv keys/ /etc/openvpn/keys && chown -R nobody:nobody /etc/openvpn/keys
```

- transférer (par SSH) toutes les clés des clients dans leurs dossier `/etc/openvpn/keys` et les donner à `nobody:nobody` (ou `nobody:nogroup` sous Debian) en 600. Cela comprend les clés suivantes :
 - `ca.crt`
 - `ta.key`
 - `dh1024.pem`
 - `client.key` et `client.crt`

5. Et si je met une passphrase pour crypter mes clés

Cela est possible mais dans ce cas vous devriez utiliser la commande `askpass` dans les fichiers de configuration. Il y a deux solutions :

- `askpass` : demande le mot de passe sur `stdin` avant de ce daemoniser : nécessite de la taper au moment du démarrage du serveur OpenVPN. Ceci peut être utile pour le client.
- `askpass fichier` : va lire le mot de passe dans le fichier `fichier`. Ce fichier devrait n'être lisible que par son propriétaire (ou `nobody` si le serveur tourne sous cet utilisateur). Cela peut être utile pour le serveur en cas de redémarrage à distance. Fichier doit contenir le mot de passe dans sa première ligne. Néanmoins cela retire une partie de l'intérêt de crypter ses clés si c'est pour laisser le mot de passe dans un fichier.

6. Configuration du serveur en mode Routed

Dans le fichier `/etc/openvpn/server.conf` :

```
#socket en écoute sur IP_public:numero_port en UDP
local IP_ou_nom_DNS_public
port 1194          #numéro de port enregistré à l'IANA
proto udp         #ou tcp
#précise la connexion Routed (/dev/tun)
dev tun
#serveur
mode server
tls-server
#taille max des paquets passant par l'interface /dev/tun
tun-mtu 1500
mssfix
#fichier contenant le certificat de la CA créée
ca keys/ca.crt
#fichier contenant la clé privée et le certificat du serveur
cert keys/nom_serveur.crt
key keys/nom_serveur.key # This file should be kept secret (600)
#paramètres Diffie-Hellman
dh keys/dh1024.pem
#fichier pour empêcher le man-in-the-middle
tls-auth keys/ta.key 0 # This file is secret, 0 = serveur

#adresse du réseau VPN dans lequel vont se trouver les clients et
le serveur
#ce réseau ne doit ni être ceux autour du client ni du serveur
#le serveur prendra la première adresse libre de la plage du pool
server adresse_reseau_vpn masque_reseau_vpn
# ou plage des adresses clients dans le VPN
#ifconfig-pool première_adresse_IP_allouable
dernière_adresse_IP_allouable

#ajoute la route au serveur pour qu'il puisse accéder au VPN
route adresse_reseau_vpn masque_reseau_vpn
#ajoute la route au client pour qu'il puisse accéder au VPN
push "route adresse_reseau_vpn masque_reseau_vpn"
```

```

#ajoute une route au client pour qu'il puisse accéder à un réseau
# se trouvant derrière le serveur
push "route adresse_réseau_serveur masque_reseau_serveur"

#autorise les clients à se voir entre eux sur le VPN
client-to-client

keepalive 10 120
cipher BF-CBC
comp-lzo

#nombre maximum de clients sur le VPN
max-clients nb_max_client

#utilisateur et groupe
user nobody
group nobody #ou nogroup sous debian

#jail dans /usr/local/openvpn
#chroot /usr/local/openvpn
#les logs (mkdir /var/log/openvpn)
status /var/log/openvpn/status.log
log-append /var/log/openvpn/openvpn.log

#niveau de log pas encore trop bavard
verb 4
#ne pas répéter les messages de logs identiques plus de 10 fois
mute 10

#si l'interface d'écoute à une IP dynamique :
#float
#persist-tun #ne pas fermer le tunnel en cas de changement IP
#persist-remote-ip #ne pas changer l'IP des clients
#persist-key #le serveur ne relie pas les clés si nobody

```

7. Configuration du serveur en mode Bridged

a) Script de démarrage supplémentaires

Si vous souhaitez utiliser un serveur OpenVPN et bridger votre interface locale, un script de bridging peut vous être utile. Il faut d'abord installer brctl.

Fichier /etc/init.d/bridge-start :

```

#!/bin/bash

#####
# Set up Ethernet bridge on Linux
# Requires: bridge-utils
#####

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged,
# for example tap="tap0 tap1 tap2".

```

```

tap="tap0"

# Define physical ethernet interface to be bridged
# with TAP interface(s) above.
eth="interface_eth_à_bridger"
eth_ip="IP_interface_à_bridger"
eth_netmask="masque_interface_à_bridger"
eth_broadcast="adresse_broadcast_interface_à_bridger"

# construit des tunnels persistants pour que brctl ne perde pas ses TAP quand
# openvpn redémarre les tunnels (restart par inactivité...)
for t in $tap; do
    openvpn --mktun --dev $t
done

#crée l'interface du bridge
brctl addbr $br
#ajoute l'interface physique Ethernet au bridge
# (pour ne pas perdre les entrées sorties autres que le VPN)
brctl addif $br $eth

#ajoute les TAPs au bridge (pour faire sortir le tunnel VPN sur le réseau)
for t in $tap; do
    brctl addif $br $t
done

# c'est l'interface du bridge $br
# qui fait le routing des paquets reçus sur la carte réseau
# à la place de celle-ci (mode promiscious)

#configure les TAPs pour écouter (et recevoir) tous les paquets
# et faire le tri de façon logicielle :
# cela permet d'avoir autant d'IPs que l'on veut sur la même carte Ethernet
for t in $tap; do
    ifconfig $t 0.0.0.0 promisc up
done

#configure l'interface physique Ethernet du bridge pour écouter tous les paquets
# et faire le tri de façon logicielle
ifconfig $eth 0.0.0.0 promisc up

#configure le bridge pour se mettre à la place
# de l'interface Ethernet physique bridgée
ifconfig $br $eth_ip netmask $eth_netmask broadcast $eth_broadcast

```

Fichier /etc/init.d/bridge-stop :

```

#!/bin/bash

#####
# Tear Down Ethernet bridge on Linux
#####

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged together
tap="tap0"

#désactiver l'interface du bridge
ifconfig $br down
#supprimer l'interface du bridge dans le logiciel de bridging
brctl delbr $br

```

```
#retirer les TAPs persistant puisque l'on arrête le bridge
for t in $tap; do
    openvpn --rmtun --dev $t
done
```

#note : cela ne remet pas en place l'interface physique Ethernet bridgée

Faire un `chmod 755` sur `/etc/init.d/bridge-st*`. Si vous souhaitez le démarrer automatiquement, c'est même le principe qu'avec le fichier `/etc/init.d/openvpn`, faites des liens symboliques dans `/etc/rc.d/rc3.d` ou `/etc/rc.d/rc5.d`.

On lance d'abord le bridge puis openvpn. On stoppe d'abord openvpn puis le bridge.

Attention : l'interface bridgée ne doit pas être l'interface de connexion des clients au serveur Vpn (par exemple sur le port 5000) sans quoi vous tourneriez en rond avec les routes mise en place. Ce ne doit pas être un interface de connexion directe à Internet car cela causerait un énorme trou de sécurité.

b) Configuration du serveur

Pour un serveur bridgé, voici une configuration.

Fichier `/etc/openvpn/server.conf` :

```
#écoute sur IP_publice_serveur:1194 en UDP
local IP_publice_serveur
port 1194
proto udp

#mode bridgé
dev tap0 #il vaut mieux indiquer tap0 que tap car sinon ca marche moins bien

#serveur
mode server

tls-server
tun-mtu 1500
mssfix
persist-key
persist-tun

#les clés et certificats
ca keys/ca.crt
cert keys/certificat-serveur.crt
key keys/certificat-serveur.key # This file should be kept secret
dh keys/dh1024.pem
tls-auth keys/ta.key 0 # This file is secret too, 0 = serveur

#adresse du bridging
#la première et la dernière IP devrait être les adresses de début
# et de fin d'un sous réseau (par ex: x.y.z.129 x.y.z.254)
# afin de simplifier le filtrage IPTABLES des clients
# (par ex: -s/-d x.y.z.128/25)
server-bridge IP_serveur_sur_reseau_bridge masque_reseau_bridge
premiere_IP_allouable_dans_res_bridge derniere_IP_allouable_dans_res_bridge

#les clients se voient entre eux
client-to-client
keepalive 10 120
cipher BF-CBC
comp-lzo

#nombre max de clients qui peuvent se connecter
max-clients 15
```

```
#utilisateur sous lequel tourne OpenVPN
user nobody
group nobody

#les logs
status /var/log/openvpn/status.log
log-append /var/log/openvpn/openvpn.log

verb 4
mute 10
```

8. Configuration du client

a) Configuration du client sous Linux

L'installation se fait comme pour le serveur.

b) Configuration du client sous Windows

Il vous faut télécharger :

<http://prdownloads.sourceforge.net/openvpn/> et l'installer

Vous devez ensuite mettre un fichier `openvpn.ovpn` (contenant la configuration du client), les fichiers `ta.key`, `ca.crt`, `nom_client.key` et `nom_client.crt` (générer avec le script `build-key` et la clé `ca.key`) dans le répertoire `config`. (Généralement `c:\program files\openvpn\config`). Lors que le VPN est lancé, vous devriez voir s'activer une seconde connexion réseau sur l'interface TAP-Win32.

c) Configuration en mode Routed

```
#indique le mode client
client
#indique le mode routed
dev tun
#indique le protocole de connexion au serveur
proto udp #ou tcp

#le serveur auquel on se connecte
remote nom_DNS_ou_IP_serveur port_serveur
#essayer indéfiniment de résoudre le nom DNS du serveur (par ex IP dynamique)
resolv-retry infinite

#choisir dynamiquement le port côté client
nobind

tls-client

#préserve les états entre deux connexions
persist-key
persist-tun

#le fichier du certificat de la CA
ca keys/ca.crt
#le certificat du client
cert keys/nom_client.crt
#la clé privée du client (600)
```

```
key keys/nom_client.key

ns-cert-type server
tls-auth keys/ta.key 1 #1 = client
cipher BF-CBC

#oblige le client à demander sa configuration au serveur
pull

#compression LZO
comp-lzo

verb 2
mute 5
```

d) Configuration en mode Bridged

```
#indique le mode client
client
#indique le mode bridged
dev tap
#indique le protocole de connexion au serveur
proto udp #ou tcp

#le serveur auquel on se connecte
remote nom_DNS_ou_IP_serveur port_serveur
#essayer indéfiniment de résoudre le nom DNS du serveur (par ex IP dynamique)
resolv-retry infinite

#choisir dynamiquement le port côté client
nobind

tls-client

#préserve les états entre deux connexions
persist-key
persist-tun

#le fichier du certificat de la CA
ca keys/ca.crt
#le certificat du client
cert keys/nom_client.crt
#la clé privée du client (600)
key keys/nom_client.key

ns-cert-type server
tls-auth keys/ta.key 1 #1 = client
cipher BF-CBC

#oblige le client à demander sa configuration au serveur
pull

#compression LZO
comp-lzo

verb 2
mute 5
```

Le client a donc besoin de son fichier de config, du fichier ta.key, du certificat de la CA (ca.crt), de sa clef privé (*nom_client.key*) et de son certificat (*nom_client.crt*).

Il faut les copier dans *c:\program files\openvpn\config* si pour le fichier de configuration ci-dessus.

9. Utilisation

- Sous Linux :

```
/etc/init.d/openvpn start ou /etc/init.d/openvpn stop
```

- Sous Windows :

double cliquer sur le fichier de configuration .ovpn.

10. Pare-feu serveur

Il faut d'abord autoriser les clients à se connecter au serveur en UDP ou TCP :

```
iptables -A INPUT -i interface_publicue -p udp --dport 1194 -j ACCEPT
iptables -A OUTPUT -o interface_publicue -p udp --sport 1194 -j ACCEPT
```

a) Mode routed

i. Connexion sur le serveur

```
# autorise les connexions VPN à faire des connexions sur le serveur VPN
# un service TCP sur le port n
iptables -A INPUT -p tcp --dport n -d première_IP_réseau_VPN -s réseau_VPN -i tun+ -j ACCEPT
iptables -A OUTPUT -p tcp --sport n -s première_IP_réseau_VPN -d réseau_VPN -o tun+ -j ACCEPT

# ping
iptables -A INPUT -p icmp -i tun+ -d première_IP_réseau_VPN -s réseau_VPN -j ACCEPT
iptables -A OUTPUT -p icmp -o tun+ -s première_IP_réseau_VPN -d réseau_VPN -j ACCEPT
```

ii. Connexion au réseau interne derrière le serveur

```
# autorise les connexions VPN à traverser le serveur VPN
# un service TCP sur le port n
iptables -A FORWARD -p tcp --dport n -i tun+ -d réseau_interne -s réseau_VPN -j ACCEPT
iptables -A FORWARD -p tcp --sport n -o tun+ -s réseau_interne -d réseau_VPN -j ACCEPT

# ping
```

```

iptables -A FORWARD -p icmp -i tun+ -d réseau_interne -s réseau_VPN -j ACCEPT
iptables -A FORWARD -p icmp -o tun+ -s réseau_interne -d réseau_VPN -j ACCEPT

# effectuer la NAT des paquets traversant
iptables -t nat -A POSTROUTING -s adresse_reseau_VPN -d adresse_reseau_interne
-o interface_reseau_interne -j SNAT --to-source IP_interne

#active le forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

```

De plus, vous devez configurer un client ou le serveur pour servir de passerelle pour accéder au réseau qui est autour de lui. Pour cela, il faut autoriser l'IP forwarding :

- Pour Windows 2000/XP, voir [cet article de la KB](#). De plus, il faut vérifier que le pare-feu ne bloque pas les paquets passant par la machine.
- Pour Linux
 - il faut activer l'IP forwarding :

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```
 - il faut aussi activer la NAT (POSTROUTING) des paquets comme indiqué précédemment

b) Mode bridged

i. Si vous utilisez un noyau 2.4

<http://ebtables.sourceforge.net/>

ii. Règles de filtrage

Il faut faire le filtrage sur les périphériques physiques. On sait, par exemple, que pour aller du réseau interne au VPN, on va de *interf_res_interne* à tap0 et inversement. Cela permet de de filtrer par *phys-in* et *phys-out*.

```

#autorise le trafic dans les deux sens : clients <--> machines res interne
iptables -A FORWARD -m physdev --physdev-in interf_res_interne --physdev-out
tap0 -j ACCEPT
iptables -A FORWARD -m physdev --physdev-in tap0 --physdev-out
interf_res_interne -j ACCEPT

```

```

#autorise le ping des machines internes par les clients mais pas inverse
iptables -A FORWARD -p icmp --icmp-type echo-reply -d plage_IP_client -m physdev
--physdev-in interf_res_interne --physdev-out tap0 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type echo-request -s plage_IP_client -m
physdev --physdev-in tap0 --physdev-out interf_res_interne -j
ACCEPT

```

```

#autorise le telnet sur les machines internes par les clients mais pas inverse
iptables -A FORWARD -p tcp --sport 23 -d plage_IP_client -m physdev --physdev-in
interf_res_interne --physdev-out tap0 -j ACCEPT

```

```
iptables -A FORWARD -p tcp --dport 23 -s plage_IP_client -m physdev --physdev-in tap0 --physdev-out interf_res_interne -j ACCEPT
```

```
#autorise le ping sur le serveur VPN par les clients mais pas inverse
iptables -A OUTPUT -p icmp --icmp-type echo-reply -d plage_IP_client -m physdev --physdev-out tap0 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -s plage_IP_client -m physdev --physdev-in tap0 -j ACCEPT
```

```
#autorise les machines du réseau interne à se connecter sur 80 du serveur
iptables -A OUTPUT -p tcp --sport 80 -d plage_IP_res_interne -m physdev --physdev-out interf_res_interne -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -s plage_IP_res_interne -m physdev --physdev-in interf_res_interne -j ACCEPT
```

c) Pare-feu client

On peut partir du principe que l'interface du VPN est relativement sûr, ce qui n'est toutefois pas toujours vrai.

Dans tous les cas, il faut s'autoriser à se connecter au serveur OpenVPN sur le port 1194 sur lequel est véhiculé le canal sécurisé.

```
#autoriser à se connecter au serveur OpenVPN
iptables -A INPUT -p udp --sport 1194 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p udp --dport 1194 -j ACCEPT
```

Dans le cas d'un VPN routé, on pourra mettre en place les règles suivantes :

```
#autoriser les paquets à passer dans le tunnel routé
iptables -A INPUT -i tun+ -j ACCEPT
iptables -A OUTPUT -o tun+ -j ACCEPT
```

Dans le cas d'un VPN bridgé, on pourra mettre en place les règles suivantes :

```
#ou autoriser les paquets à passer dans le tunnel bridgé
iptables -A INPUT -i tap+ -j ACCEPT
iptables -A OUTPUT -o tap+ -j ACCEPT
```

Si l'on veut peut être plus restrictif et n'autoriser que les logiciels clients sur la machine client et ne pas autoriser les machines du réseau interne du serveur à se connecter en SSH (par exemple) sur le client, on pourra ajouter `-m state --state RELATED,ESTABLISHED` au règles précédentes pour INPUT sur tun+ ou tap+.

d) Règles à éviter

```
iptables -A INPUT -f -j DROP
```

Elles droppent les paquets fragmentés, ce qui peut être TRES problématiques dans le cadre d'un VPN...

11. Considération de sécurité

Le seul problème d'OpenVPN est que si un client se fait voler sa clé privée, le voleur peut alors se connecter au VPN et accéder à toutes les machines internes à celui-ci. Il est donc nécessaire que les

clients du VPN soient muni (pour les Windows) d'un antivirus et antispyware afin de ne pas se faire subtiliser ses clés privées et d'éviter la propagation rapide d'un virus à toutes les machines du VPN. De plus, un filtrage/pare-feu des clients et du serveur est nécessaire afin de ne pas laisser la possibilité à un vers de passer dans tous les sens.

IV. Annexe : le protocole SSL et OpenSSL

1. Qu'est-ce qu'OpenSSL ?

SSL a été à l'origine développé par Netscape.

OpenSSL est un version libre du protocole SSL (Secure Sockets Layer version 1 et 2) et TLS (Transport Layer Security = SSL version 3). Il permet de crypter toutes les données échangées entre le client et le serveur de façon à ce que seul le serveur puisse décrypter ce qui vient du client et inversement. Un éventuel pirate ne peut pas, dans un temps raisonnable, décrypter les informations.

SSL sert de support :

- à SSH pour donner un telnet sécurisé et faire des tunnels cryptés simplement
- à HTTP pour sécurisé les sites de Web marchands
- à POP ou IMAP pour sécurisé la récupération de mail
- à tout autre protocole en clair afin de le sécuriser

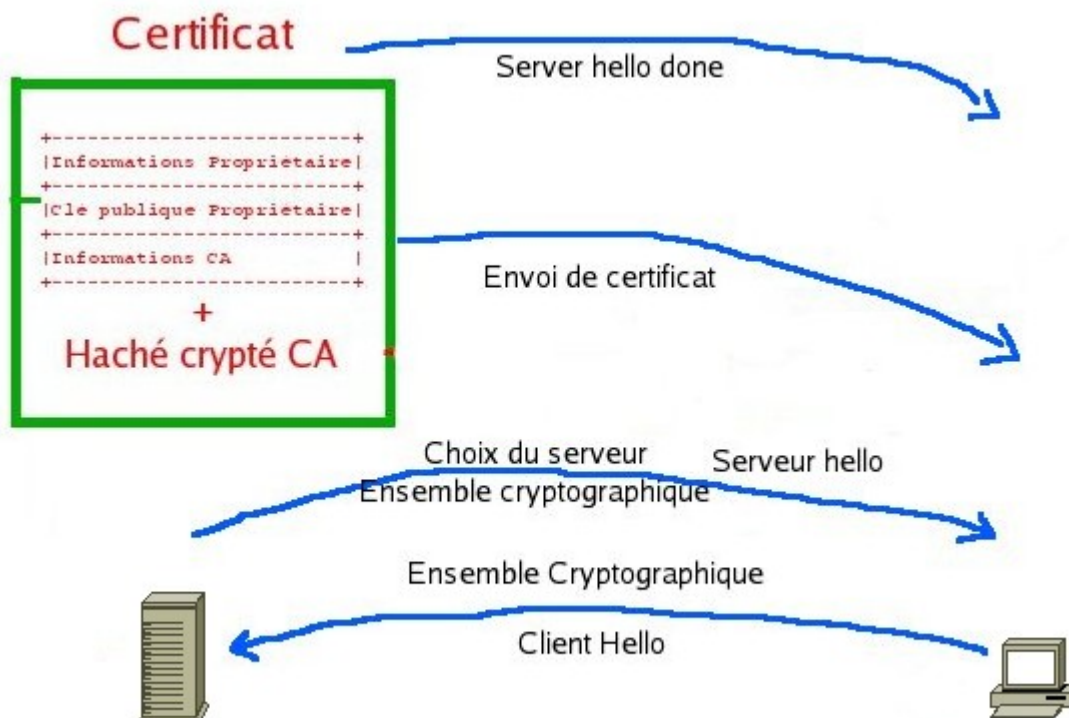
SSL a trois buts :

- confidentialité des échanges grâce au cryptage symétrique
- intégrité des données grâce au fonction de hachage
- authentification des entités communicantes grâce aux certificats

2. Connexion SSL

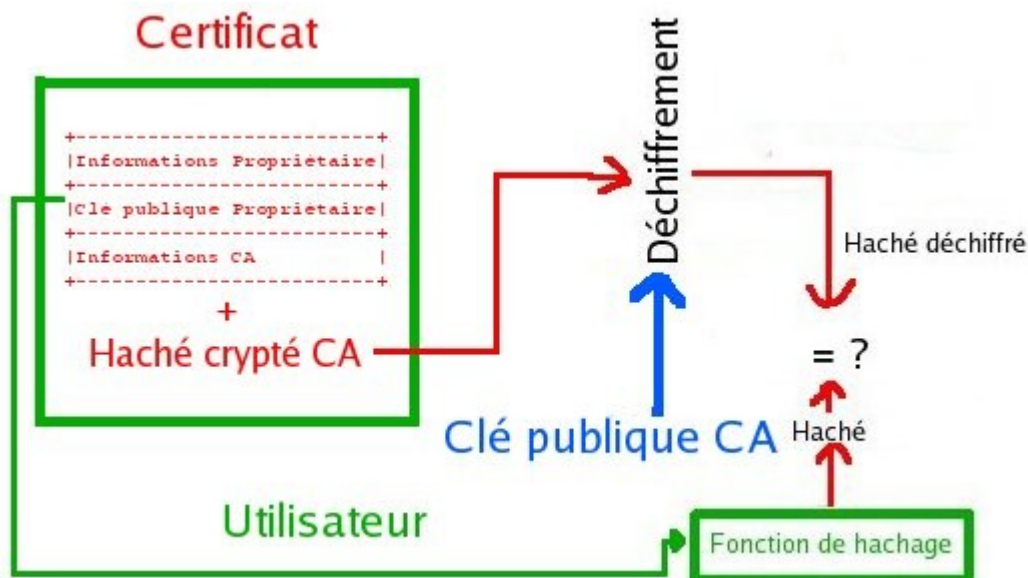
Les étapes d'une connexion SSL sont les suivantes (du moins) :

1. établissement de la connexion : choix de la méthode de cryptage

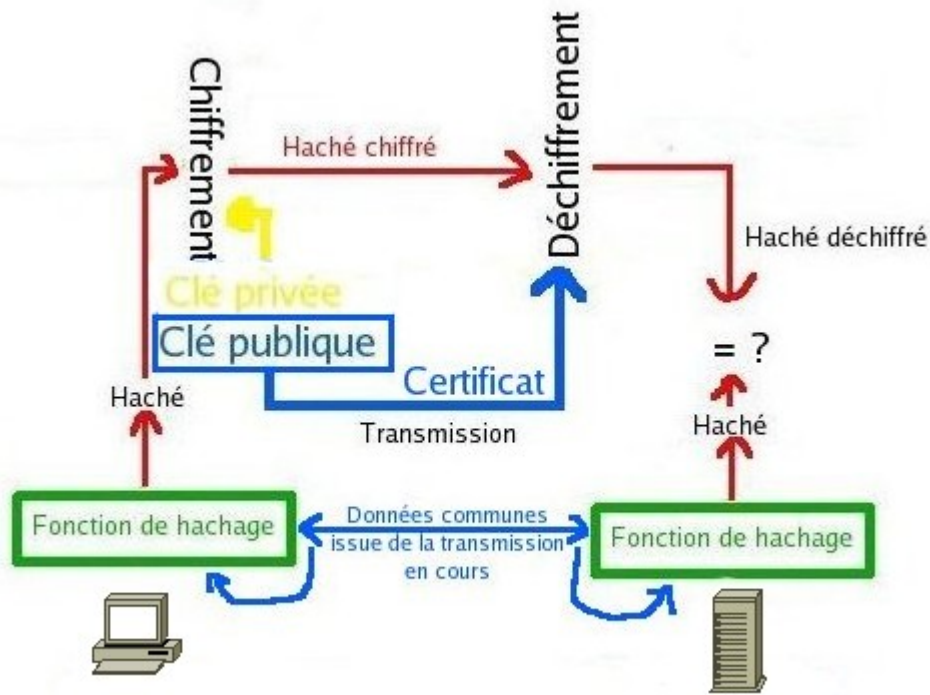


- le client envoie un « client-hello » au serveur qui comprend :

- la version de SSL utilisée par le client
 - un id de session aléatoire
 - un ensemble cryptographique que le serveur devrait pouvoir supporter :
 - une méthode de cryptage à clé publique (asymétrique) pour le transfert des clés de session
 - une méthode de cryptage à clé privée (symétrique) (utilisera la clé de session) pour le transfert des données après l'établissement de la connexion (aucune, RC4(40bits), RC4(128bits), RC2(40bits), DES(40bits), DES(56bits), 3DES(168bits), IDEA (128bits) ou FORTEZZA(96bits))
 - une méthode de hachage pour calculer un digest des paquets transmis pendant la connexion, évite les attaques par relai (aucune, MD5(128bits) ou SHA-1(160bits)).
 - une méthode de compression
 - le serveur répond un « serveur-hello » au client (ou un code d'erreur = abandon de la connexion) contenant l'ensemble cryptographique qu'il a choisi sous la forme SSL_X_WITH_Y_Z :
 - la méthode de cryptage symétrique X
 - la méthode de cryptage asymétrique Y
 - la méthode de hachage Z
 - (la méthode de compression)
 - le serveur envoie son certificat et éventuellement une demande du certificat client (qui contient une liste de type de certificat et de CA autorisés par le serveur)
 - le serveur envoie un « server hello done »
1. authentification du serveur

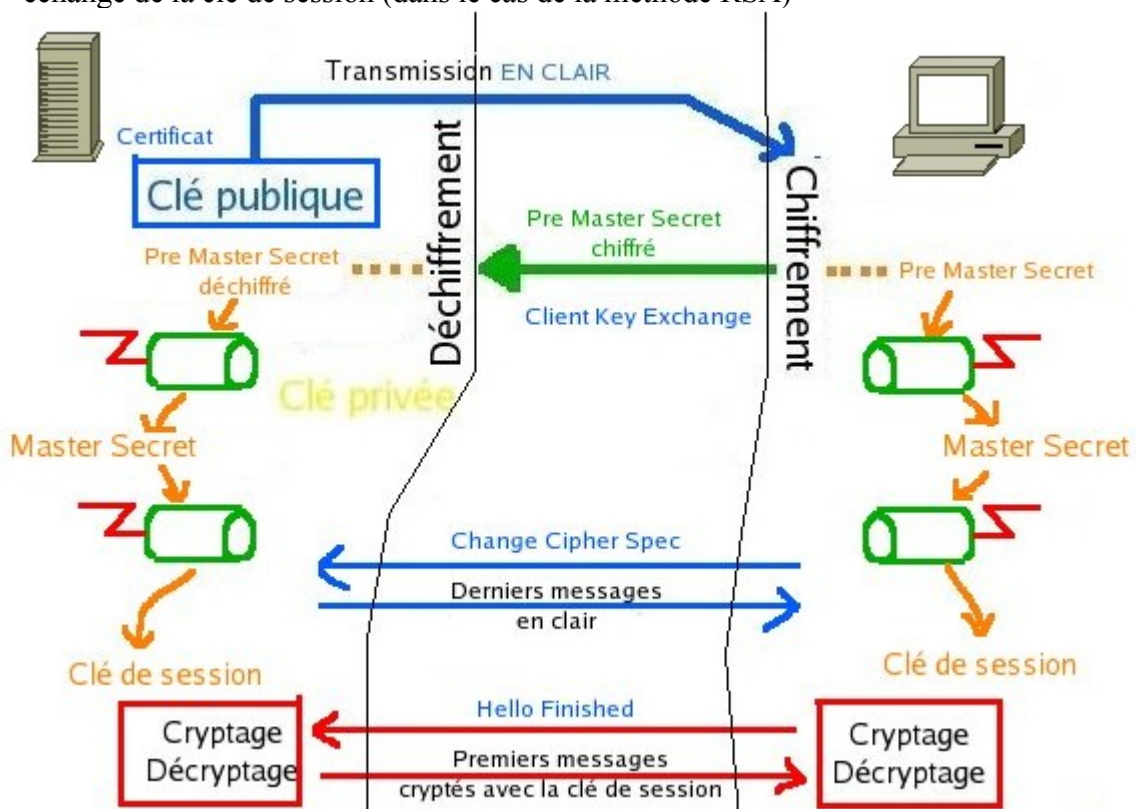


- une fois que le client reçoit le message « server hello done », il vérifie le certificat de la façon suivante :
 - trouver le nom de l'algorithme de chiffrement utilisé pour la signature du certificat dans celui-ci
 - calculer le digest du certificat
 - décrypter la signature présente dans le certificat avec l'algorithme trouvé et la clé publique de la CA **qu'il possède au préalable**
 - comparer le digest calculé et le digest décrypté : s'ils sont égaux, la connexion est établie sinon erreur
1. authentification facultative du client (si le serveur est configuré pour)



- le client scelle un morceau de donnée connu du client et du serveur avec la clé privée de son certificat puis envoie ce sceau et son certificat
- le serveur essaie de décrypter le sceau avec la clé publique du client trouvée dans son certificat, pour voir si les données correspondent à celle qu'il connaît. Si oui, la connexion continue, sinon annulation.

1. échange de la clé de session (dans le cas de la méthode RSA)



- le client envoie le « client key exchange » contenant le Pre Master Secret, un nombre aléatoire servant à générer le Master Secret, crypté avec la clé publique du serveur (se trouvant dans le certificat)

- le serveur décrypte le Pre Master Secret avec sa clé privée
 - le client et le serveur calcule le Master Secret avec une série cryptographique qui va servir à calculer la clé de session
 - le client et le serveur calcule la clé de session (symétrique) à l'aide du Master Secret.
 - le client envoie le message « change cipher spec » afin de signaler la mise en place de la clé de session
 - le client envoie un message de fin crypté avec la clé de session
 - le serveur envoie le message « change cipher spec » afin de signaler la mise en place de la clé de session
 - le serveur envoie un message de fin crypté avec la clé de session
1. échanges cryptés
 - maintenant, les échanges sont cryptés dans les deux sens
 - il peut arriver que le processus de génération de clé de session recommence au hasard.

V. Bibliographie

Ethernet-Bridging :

<http://www.tldp.org/HOWTO/Ethernet-Bridge-netfilter-HOWTO.html>

<http://www.tldp.org/HOWTO/BRIDGE-STP-HOWTO/index.html>

OpenVPN :

[OpenVPN - An Open Source SSL VPN Solution by James Yonan](#)

[OpenVPN 2.0 HOWTO](#)

[OpenVPN 2 HOWTO français](#)

[Installation d'une passerelle Linux](#)