OpenVPN

Installation

Pour l'installation, rien de plus simple. Nous l'effectuons depuis les dépôts officiel Debian. On s'assure cependant que notre liste de paquets est à jour :

```
# apt-get update && apt-get install openvpn
```

Configuration

Easy RSA

On commence la configuration avec **Easy RSA**, installé en même temps que notre **OpenVPN**. On créer le dossier puis l'on copie ce dont nous avons besoin avant de nous rendre dans notre nouveau dossier :

```
# mkdir /etc/openvpn/easy-rsa/
# cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0/* /etc/openvpn/easy-
rsa/
# cd /etc/openvpn/easy-rsa
```

On va maintenant éditer le fichier *vars*

```
15 | # export EASY_RSA="`pwd`"
   | export EASY_RSA="/etc/openvpn/easy-rsa"

53 | # export KEY_SIZE=1024
   | export KEY_SIZE=4096
```

Les informations des lignes 64 à la fin du fichier peuvent être modifier pour gagner du temps par la suite.

Le serveur

On passe à la configuration du certificat avec les clefs qui serviront aux échanges chiffrés entre les clients et le VPN. Nous nous trouvons toujours dans le dossier **/etc/openvpn/easy-rsa**

```
source ./vars
./clean-all
./build-ca
./build-key-server <nomDuServeur>
```

La commande *clean-all* permet de nettoyer l'ensemble des fichiers présent.

Common Name doit être identique entre le certificat et la clé serveur.
Challenge Password reste vide.

On oubliera pas la sécurité, on va faire appel à l'échange de clés de Diffie-Hellman pour protéger les échanges entre le(s) client(s) et le serveur.

```
cd ../
./build-dh
```

On activera ensuite une première protection contre, notamment, les attaques de type DOS grâce à l'outils HMAC (*Hash-based Message Authentication Code*).
Si un client demande à s'authentifier au serveur sans entamer le dialogue par l'envoi de ce HMAC, OpenVPN ne va même pas entamer la procédure et donc limiter la surcharge serveur.

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

Le client

On à maintenant tout ce qu'il nous faut coter serveur. Passons maintenant au client.

```
./build-key-pass <nomDuClient>
```

On ajoute ensuite en couche de sécurité grâce à l'algorithme 3DES via la commande

```
cd keys
openssl rsa -in <nomDuClient>.key -des3 -out <nomDuClient>.3des.key
```

On répète cette étape pour chacun des clients que l'on souhaite créer.
On peut également la rendre plus simple grâce à un petit script auquel on passe le nom du client en paramètre et que l'on place dans notre dossier OpenVPN.

```
#!/bin/bash

if [ -d "easy-rsa/keys/$1" ]; then
 echo "[ERREUR] L'utilisateur existe déjà"
 exit
fi

./easy-rsa/build-key-pass $1
openssl rsa -in easy-rsa/keys/$1.key -des3 -out easy-rsa/keys/$1.3des.key
```

La machine

On doit maintenant s'occuper du routage. En effet, pour le moment nos clients n'ont pas la possibilité d'accéder au réseau public. Pour rendre cette action possible, nous allons utiliser les possibilités de routage de linux en exècutant la commande suivante, pour donner le droit de router à chaud :

```
sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
```

Pour que ce réglage soit persistant il faut l'indiquer dans le fichier de configuration **/etc/sysctl.conf** en dé-commentant la ligne **net.ipv4.ip_forward = 1**

Il ne nous reste que les règles de routage à faire

```
iptables -I FORWARD -i tun0 -j ACCEPT
iptables -I FORWARD -o tun0 -j ACCEPT
iptables -I OUTPUT -o tun0 -j ACCEPT
iptables -A FORWARD -i tun0 -o venet0 -j ACCEPT
iptables -t nat -A POSTROUTING -o venet0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 10.8.6.0/24 -o venet0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 10.8.6.2/24 -o venet0 -j MASQUERADE
```

Fichier serveur

Maintenant que nous avons tout préparé, on passe à la création de notre serveur. On retourne dans notre dossier OpenVPN.
A cause de restriction stricte sur le réseau, nous plaçons notre serveur sur le port 443 (HTTPS). On garde quand même notre serveur Web accessible. Pour cela, on le met à l'écoute sur le port 444 (choix arbitraire) et on indique à OpenVPN de transférer toute les requêtes qui ne lui sont pas destinée vers ce port.

```
port 443
port-share 127.0.0.1 444
proto tcp
tcp-nodelay
dev tun
tun-mtu 1200
;fragment 1300
;mssfix
mtu-disc yes
comp-lzo
persist-key
persist-tun
keepalive 10 20
server 10.8.6.0 255.255.255.0
client-to-client

push "redirect-gateway bypass-dhcp"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"

ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/<nomDuServeur>.crt
key /etc/openvpn/easy-rsa/keys/<nomDuServeur>.key
dh /etc/openvpn/easy-rsa/keys/dh4096.pem
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
cipher AES-256-CBC

status openvpn-status.log
```

```
verb 3
```

Il peut arriver qu'OpenVPN affiche le message d'erreur lors du démarrage l'empêchant d'aboutir

```
Note: Cannot open TUN/TAP dev /dev/net/tun: No such device (errno=19)
Note: Attempting fallback to kernel 2.2 TUN/TAP interface
Cannot open TUN/TAP dev /dev/tap0: No such file or directory (errno=2)
Exiting
```

Pour palier à ce problème, nous exécutons les deux commandes ci-dessous

```
# mkdir /dev/net
# mknod /dev/net/tun c 10 200
```

Fichier client

Pour nous facilité la tâche, nous faisons appel à un petit script maison qui se chargera de tout ! Mais juste avant, nous procédons à la création d'une *base type* pour tout nos clients

```
client
dev tun
proto tcp-client
remote <ipDuServeur> 443
resolv-retry infinite
nobind
redirect-gateway def1

ns-cert-type server
key-direction 1
cipher AES-256-CBC

persist-key
persist-tun

comp-lzo

verb 3
```

Nous plaçons cette base dans **/etc/openvpn/easy-rsa/keys/Default.txt** avant de revenir dans notre dossier OpenVPN pour y mettre notre script qui prendra lui-aussi le nom du client en paramètre.

```
#!/bin/bash

# Déclaration des variables par défaut
NAME=$1
DEFAULT="easy-rsa/keys/Default.txt"
FILEEXT=".ovpn"
CRT="easy-rsa/keys/$NAME.crt"
KEY="easy-rsa/keys/$NAME.3des.key"
CA="easy-rsa/keys/ca.crt"
```

```
TA="easy-rsa/keys/ta.key"

# On vérifie que la clé publique associé au nom du client existe
if [ ! -f $CRT ]; then
 echo "[ERREUR] Clé publique non trouvée"
 exit
fi
echo "Clé publique trouvée"

# On vérifie qu'une clé privée existe pour ce client
if [ ! -f $KEY ]; then
 echo "[ERREUR] Clé privée 3DES non trouvée : $KEY"
 exit
fi
echo "Clé privée trouvée : $KEY"

# On vérifie l'existence de la clé CA
if [ ! -f $CA ]; then
 echo "[ERREUR]: Clé publique CA non trouvée : $CA"
 exit
fi
echo "Clé publique CA trouvée : $CA"

# On vérifie l'existence de la clé tls-auth
if [ ! -f $TA ]; then
 echo "[ERREUR]: Clé privée tls-auth non trouvée : $TA"
 exit
fi
echo "Clé privée tls-auth trouvée : $TA"

# Prêt à générer le fichier de configuration
# On commence par insérer le fichier de base
cat $DEFAULT > $NAME$FILEEXT

# On ajoute la clé publique du CA
echo "<ca>" >> $NAME$FILEEXT
cat $CA >> $NAME$FILEEXT
echo "</ca>" >> $NAME$FILEEXT

# On ajoute la clé publique du client
echo "<cert>" >> $NAME$FILEEXT
cat $CRT | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >>
$NAME$FILEEXT
echo "</cert>" >> $NAME$FILEEXT

# On ajoute la clé privée du client
echo "<key>" >> $NAME$FILEEXT
cat $KEY >> $NAME$FILEEXT
echo "</key>" >> $NAME$FILEEXT

# Enfin, on ajoute la clé tls-auth
```

```
echo "<tls-auth>" >> $NAME$FILEEXT
cat $TA >> $NAME$FILEEXT
echo "</tls-auth>" >> $NAME$FILEEXT

# On range tout les fichiers de l'utilisateur dans un sous-dossier à son nom
mkdir easy-rsa/keys/$NAME
mv easy-rsa/keys/$NAME.* easy-rsa/keys/$NAME/
mv $NAME.ovpn easy-rsa/keys/$NAME/

echo "Terminé ! $NAME$FILEEXT généré avec succès."
```

On aura donc deux scripts à lancer pour créer un utilisateur. Le premier serveur à générer la clé et le second le fichier de configuration client.