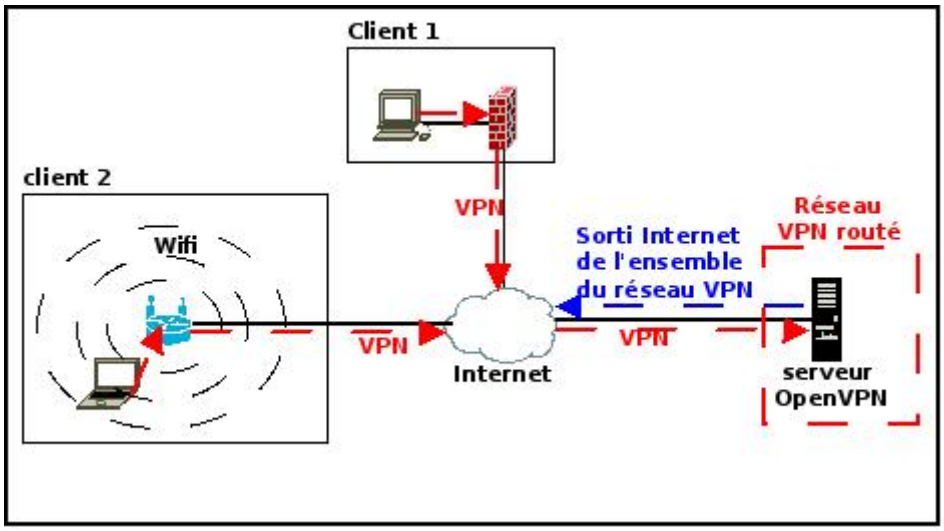
# OpenVPN : configuration tun avec authentification PAM

Il y a tellement de configuration différente d'OpenVPN, selon les besoins, je vous présente une configuration le mode TAP(routé). Qui peut être intéressante, en plus il y a une partie authentification qui se fait directement sur le serveur avec le module PAM, que j'ai ajouté.

Voici, le schéma de principe suivant




Avec c'est article vous obtiendrez un réseau sécurisé, votre sorti Internet sera celle de votre serveur OpenVPN et les différents clients pourront se voir.

Pour l'installation, la génération et gestion des certificats, je vous invite ici.

## Création du fichier de configuration serveur :

```
mode server #pas de surprise
port 443    #par défaut le port 1194, mais pour passer dans les hôtels,
proxy, ..., c'est mieux
proto tcp   #par défaut c'est udp, mais pour la raison du dessus mettre le
protocole TCP, pour faire genre on surf en https
dev tun     #il existe 2 type de VPN tun (en gros mode router) et tap (en
gros mode pont)

ca ca.crt
cert server.crt
key server.key  # ce fichier doit rester secret c'est la clé RSA privé du
serveur
dh dh.pem

server 10.8.0.0 255.255.255.0 #pour le DHCP
ifconfig-pool-persist ipp.txt #pour pouvoir suivre les IP distribué
client-to-client              #Pour que vos clients puisse se voir
intéressant quand on veut faire une LAN, que de souvenir^^
```

```
plugin /usr/local/lib/openvpn/plugins/openvpn-plugin-auth-pam.so login #
pour activer l'authentification des utilisateurs, si c'est pas le bon chemin
pour vous faire find / -name *auth-pam.so
username-as-common-name
# Pour vérifier l'authentification, utilisez le nom d'utilisateur
authentifié comme nom commun, plutôt que le nom commun du certificat client.

keepalive 10 120  #Une directive d'aide conçu pour simplifier l'expression
de --ping et --ping-redémarrage dans des configurations en mode serveur.

cipher  AES-256-CBC        # AES

push "redirect-gateway def1 bypass-dhcp"  #permet de rediriger0 tous le
trafic des clients vers le tunnel
push "dhcp-option DNS 208.67.222.222"     #permet de pousser d'autres
serveur DNS  sur les clients
push "dhcp-option DNS 208.67.220.220"

comp-lzo  #Utiliser la compression LZO rapide

user nobody
group nogroup

persist-key
persist-tun

status /var/log/openvpn-status.log
log-append  /var/log/openvpn.log
verb 3
mute 20
```

## lancer openvpn server :

```
openvpn --daemon --writepid /var/run/openvpn.pid --cd /etc/openvpn --config
server.conf
```

Voici un script qui vous permettra de lancer le daemon ou de le stopper.

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:             openvpn
# Required-Start:       $remote_fs $syslog
# Required-Stop:        $remote_fs $syslog
# Default-Start:        2 3 4 5
# Default-Stop:
# Short-Description:    openvpn
### END INIT INFO
```

```
name=openvpn
bin=/usr/local/sbin/${name}
dir=/etc/${name}
pid=/var/run/${name}.pid
config=server.conf

start () {
        $bin --daemon --writepid $pid --cd $dir --config $config
}
stop () {
        killall -TERM $name
}
case "$1" in
        start)
                start
        ;;
        stop)
                stop
        ;;
        restart)
                stop
                start
        ;;
        *)
                echo $"Usage: ${IPTABLES} {start|stop|restart}"
        ;;
esac
```

Pour que ce script soit présent au démarrage faire :

```
update-rc.d openvpn default
```

## parti routage du serveur :

Pour commencer on va activer le `forward des packets` sous Linux :

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Pour que la modification soit prise au redémarrage du serveur :

```
vim /etc/sysctl.conf
```

Puis vous dé-commenté la ligne `net.ipv4.ip_forward` et on ajoute 1 :

```
net.ipv4.ip_forward=1
```

Et pour finir il nous faudra masquer les IPs pour que les clients puissent sortir sur l'accès Internet de votre serveur pour cela on utilise `iptables`

```
iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE
```

Pour que la modification soit prise au redémarrage du serveur, il faut donc faire un script, j'ai une méthode que certain diront, c'est compliqué, mais inspiré de ce qui existe sous RedHat pour gérer cela, j'aime .

Cela consiste à faire une sauvegarde des règle `iptables` dans par exemple `/etc` avec un nom de fichier par exemple `iptables` qui servira de fichier de configuration et pour finir on fera un script qui va gérer nos règles (start|stop|restart|save).

quand je met **stop** et **start** on ne stop pas réellement `iptables` on purge les règles ou on charge les règles `iptables`

Donc on commence par faire une sauvegarde de nos règles `iptables`

```
iptables-save > /etc/iptables
```

Et voici, mon script de gestion des règles iptables à mettre dans `/etc/init.d/iptables`:

```
#! /bin/sh

### BEGIN INIT INFO
# Provides:             iptables
# Required-Start:       $remote_fs $syslog
# Required-Stop:        $remote_fs $syslog
# Default-Start:        2 3 4 5
# Default-Stop:
# Short-Description:    iptables
### END INIT INFO


IPTABLES=iptables
IPTABLES_DATA=/etc/$IPTABLES
. /lib/lsb/init-functions

if [ ! -x /sbin/$IPTABLES ]; then
        log_daemon_msg "${IPTABLES}: /sbin/$IPTABLES does not exist."
        err=1
        case "$err" in
                0) log_end_msg 0;;
                *) log_end_msg 1; exit 1;;
        esac
fi

flush_n_delete() {
        for table in "filter" "nat"
        do
                #Flush firewall rules.
```

```
                $IPTABLES -t $table -F
                #Delete firewall chains.
                $IPTABLES -t $table -X
                #Set counter to zero.
                $IPTABLES -t $table -Z
        done
}
set_policy() {
        #set policy for configured tables.
        policy=$1
        for table in "filter" "nat"
        do
                if [ $table = "filter" ]
                then
                        for chain_filter in "INPUT" "OUTPUT" "FORWARD"
                        do
                                $IPTABLES -t $table -P $chain_filter $policy
                        done
                fi
                if [ $table = "nat" ]
                then
                        for chain_nat in "PREROUTING" "INPUT" "OUTPUT"
"POSTROUTING"
                        do
                                $IPTABLES -t $table -P $chain_nat $policy
                        done
                fi
                if [ $table = "nat" ]
                then
                        for chain_nat in "PREROUTING" "INPUT" "OUTPUT"
"POSTROUTING"
                        do
                                $IPTABLES -t $table -P $chain_nat $policy
                        done
                fi
        done
}
start() {
        if [ ! -f "$IPTABLES_DATA" ]
        then
                log_daemon_msg "$IPTABLES_DATA is not existe"
                err=1
                case "$err" in
                        0) log_end_msg 0;;
                        *) log_end_msg 1; exit 1;;
                esac
        fi
        log_daemon_msg "load the rules and chains."
                $IPTABLES-restore < $IPTABLES_DATA
        case "$?" in
                0) log_end_msg 0;;
```

```
                *) log_end_msg 1; exit 1;;
        esac
}
stop() {
        log_daemon_msg "Set default chain policy to ACCEPT."
                set_policy ACCEPT
        case "$?" in
                0) log_end_msg 0;;
                *) log_end_msg 1; exit 1;;
        esac
        log_daemon_msg "flush the rules and delete chains."
                flush_n_delete
        case "$?" in
                0) log_end_msg 0;;
                *) log_end_msg 1; exit 1;;
        esac
}
save() {
        if [ ! -f "$IPTABLES_DATA" ]
        then
                log_daemon_msg "$IPTABLES_DATA is not existe"
                err=1
                case "$err" in
                        0) log_end_msg 0;;
                        *) log_end_msg 1;;
                esac
      fi
      log_daemon_msg "saving firewall rules to $IPTABLES_DATA"
              $IPTABLES-save > $IPTABLES_DATA
              chmod 600 $IPTABLES_DATA
      case "$?" in
              0) log_end_msg 0;;
              *) log_end_msg 1; exit 1;;
      esac

}
case "$1" in
        start)
                start
        ;;
        stop)
                stop
        ;;
        restart)
                stop
                start
        ;;
        save)
                save
        ;;
        *)
```

```
                echo $"Usage: ${IPTABLES} {start|stop|restart|save}"
        ;;
esac
```

Pour que ce script soit présent au démarrage faire :

```
update-rc.d iptables default
```

### Voici la configuration d'un client pour ce genre de VPN:

```
client
dev tun
proto tcp
remote <IP_du_serveur> 443
resolv-retry infinite
auth-nocache
remote-cert-tls server
nobind
persist-key
persist-tun
ca ca.crt
cert <client>.crt
key <client>.key
auth-user-pass
cipher BF-CBC #mettre le même algo de chiffrement logique
comp-lzo
verb 3
mute 20
```

Pour des raisons de digestion, j'ai segmenté l'article, car comme je l'ai précisé, il existe plusieurs configurations d'OpenVPN, pour la partie gestion de la PKI ici