Retour au sommaire des How-to

OpenVPN 2 HOWTO français

Préconisations, Installation, configuration.
Par Philippe Humeau (philippe.humeau@nbs-system.com) le 10/01/2005

1. Présentation

Ce HOWTO décrit l'installation d'un serveur OpenVPN et la configuration de ses clients de connexion. Les exemples sont réalisés avec un Linux (Debian/GNU Linux 3.0 Sarge Testing, noyau 2.6.10 avec patch GRsec 2.1.0) et des clients sous Windows 2000 (SP4) et XP (SP2). Il est présupposé que vous avez des connaissances de base en Unix et en réseau et que vous êtes capables d'adapter les scripts qui sont livrés dans ce HOWTO selon vos besoins.

Il n'est plus utile de rappeler l'intérêt ou le but d'un VPN. Si vous lisez ces lignes c'est que vous avez déjà ce besoin et que vous en êtes au stade de l'installation. Ce Howto ne parle donc pas (ou très peu) des bénéfices et des risques du VPN, mais exclusivement de son paramétrage et des petites astuces qui vous aideront à déployer cet outil.

OpenVPN est un système de réseau privé virtuel développé par James Yonan (jim@yonan.net) sur le protocole SSL et non sur le protocole IPSEC comme la plupart des VPN.

Le client dont l'installation est décrite dans ce Howto est écrit et maintenu par Mathias Sundman mathias@nilings.se, le wizard intégré pour générer des certificats et des clefs est réalisé par Vlada Macek tuttle@bbs.cvut.cz, la librairie LZO utilisé par OpenVPN est l'oeuvre de Markus F.X.J. Oberhumer, le driver TAP32 pour Windows est réalisé par Damion K. Wilson, et le système NSIS permettant de compiler le client est écrit et maintenu par Joost Verburg.

Tous ces composants sont sous licence GPL : http://www.gnu.org/copyleft/gpl.html

Rapidement, et pour ne pas rentrer dans une polémique concernant IPSEC, les avantages des VPN en SSL sont nombreux, notamment la possibilité d'utiliser le client de connexion même en cas de NAT. Le fait d'utiliser un protocole standard et maîtrisé (SSL) permet d'utiliser de nombreuses possibilités d'interfaçage avec d'autres composants logiciels.

Bridger ou Router ?

Il existe deux modes de fonctionnement VPN différents dans le système proposé par James Yonan, " Routed " et " Bridged ". Globalement, si vous ne souhaitez que mettre en relation entre elles des machines distantes, orientez vous vers un mode " Routed ". Si vous souhaitez mettre en relation des réseaux entre eux mettez en place un mode " Bridged ".

Le mode " Bridged " ou pont, permet de faire un pont éthernet entre deux réseaux. Les interfaces VPN et LAN sont alors liées entre elles en une seule entité et permettent de communiquer entre les sous réseaux concernés.

Si votre station cliente du VPN possède une interface en 192.168.0.4 vers son LAN et une interface virtuelle de VPN en 10.0.0.4, le serveur étant en 10.0.0.1 coté VPN et dans un LAN en 192.168.1.4, si vous bridgez les interfaces du client et du serveur, les réseaux 192.168.0.0/24 et 192.168.1.0/24 se verront entre eux. Le client aura accès au réseau 192.168.1.0/24 avec les mêmes filtrages que la machine 192.168.1.4 et de la même façon, le serveur aura accès au réseau 192.168.0.0/24.

Il existe bien sur des considérations non négligeables de sécurité à ce sujet, car la perméabilité inter réseaux peut mettre en danger les réseaux concernés si l'un des membres du VPN est compromis.

Les entreprises souhaitant interconnecter plusieurs sites (réseaux) distants et les joueurs en lignes souhaitant se faire un réseau privé opteront plus pour un réseau bridgé et les entreprises qui désirent un réseau pouvant accueillir des nomades et les filtrer de façon plus fines seront probablement plus intéressées par le mode routé.

Note sur le mode Bridge : Il n'est pas possible (ou tout à fait expérimental) de bridger des connexions sous Windows 95->2000, seul les Windows XP et 2003 supportent ce mode de fonctionnement actuellement. Après d'infructueuses tentatives sous 2000 avec le driver de bridging fournis par http://www.ntkernel.com/utilities/etherbridge.shtml, je préfère vous déconseiller cette méthode. Si toutefois vous arriviez à bridger des connexions sous Windows 2000, n'hésitez pas à m'en faire part. Un howto très visuel (mais en anglais) écrit par Adam Pavelec décrit le bridging de connexion avec Windows XP : http://www.pavelec.net/adam/openvpn/bridge/

Configuration des différentes machines utilisées dans les exemples :
Serveur : K6 II 266 Mhz, 160 Mo de RAM, deux cartes réseaux (3com 3C905 B et Intel ether express Pro 100). GNU/Linux Debian 3.0 Sarge, Testing, noyau 2.6.10-grsec, Iptables v1.2.11
OpenVPN Server 2.0_rc6.

Clients : AMD Athlon XP 2500+, 768 Mo de RAM, 3COM 3C905B, Windows 2000 SP4
Intel P4, 1 Go de RAM, Marvel yukon Gigabit, Windows XP SP2
Intel P3, 512 Mo de RAM, Intel Ether Express Pro 100, Windows XP SP2

2. Préparation du serveur

Le serveur doit valider un certain nombre de pré requis afin de pouvoir faire tourner OpenVPN correctement :

- Avoir les options noyau correctement paramétrées
- Avoir la librairie LZO installée sur le système
- Avoir la librairie OpenSSL installée (ainsi que les composants Dev)
- Avoir un les outils de bridging sous Linux (en cas de bridging, brctl de Lennert Buytenhek buytenh@gnu.org)

Si vous utilisez un Linux de type Debian, la commande :

```
apt-get install bridge-utils openssl libssl-dev liblzo1 liblzo-dev
```

Devrait s'occuper toute seule pour vous des trois derniers points. Il ne vous restera que la partie noyau à gérer. Pour les utilisateurs d'autres distributions Linux :

Récupérez et installez :

- La librairie LZO :
  http://www.oberhumer.com/opensource/lzo/download/lzo-1.08.tar.gz
  ```
  tar zxvf lzo-1.08.tar.gz
  cd lzo-1.08
  ./configure && make && make check && make test
  make install  (avec le compte root)
  ```
- La librairie openssl :
  http://www.openssl.org/source/openssl-0.9.7e.tar.gz
  ```
  tar zxvf openssl-0.9.7e.tar.gz
  ./config && make && make test && make install
  ```
- Brctl :
  http://prdownloads.sourceforge.net/bridge/bridge-utils-1.0.4.tar.gz
  ```
  tar zxvf bridge-utils-1.0.4.tar.gz
  cd bridge-utils-1.0.4
  ./configure && make && make install
  ```
- Pour les utilisateurs d'autres Unix :
  Récupérez et installez les mêmes librairies et utilitaires, certains ajustements seront probablement nécessaires.
- Pour les utilisateurs de Windows :
  Je suppose, sans l'avoir testé, que vous aurez les mêmes démarches à faire sous cygwin. (www.cygwin.com)

3. Ajout des options dans le kernel

J'utilise le kernel 2.6.10, notamment pour sa robustesse et surtout car le patch GRsec 2.1.0 est disponible pour cette version du kernel au moment de l'écriture de ce Howto.

Pour de multiples raisons trop longues à décrire ici, le patch GRsec est incontournable en terme de sécurité Linux, je vous conseil donc de le mettre en place, même si ce n'est pas du tout indispensable pour l'installation d'OpenVPN.

Si vous voulez Grsec récupérez le sur http://www.grsecurity.net/:
http://www.grsecurity.net/grsecurity-2.1.0-2.6.10-200501081640.patch (pour le 2.1.0 pour le kernel 2.6.10)

Le kernel est disponible sur : ftp.kernel.org
ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.10.tar.gz (pour le kernel 2.6.10)

Ensuite, déplacer les fichers grsecurity-2.1.0-2.6.10-200501081640.patch et linux-2.6.10.tar.gz archives dans /usr/src puis tapez :

```
tar zxvf linux-2.6.10.tar.gz
patch -p0 < grsecurity-2.1.0-2.6.10-200501081640.patch
```

Vous devriez maintenant avoir une arborescence des sources du kernel patchées avec GRsec.

Ensuite :

```
cd linux-2.6.10 && make menuconfig
```

configurez votre kernel selon les besoins de votre machine et mettez les options suivantes :

Activez les options suivantes :

```
Device Driver
   -> Networking support
      -> Networking Options
         -> Universal Tun/Tap device driver support
         -> 802.1d Ethernet Bridging

Optionnellement (mais conseillé) :
Device Driver
   -> Networking support -> Networking Options
      -> Network Packet Filtering
         -> IP Netfilter et Bridge Netfilter

(si vous avez patché avec GRsec) :
Security Options
  -> GRsecurity
     -> Adress Space Protection
        -> Deter Exploit brutforcing
     -> FileSystem protection
```

```
        -> Chroot jail restriction (tout)
     -> executable protection
        -> randomized PID
     -> Network protection
        -> (larger entropy pool, truly random tcp isn, random IP ids)
     -> sysctl support -> sysctl support
  -> PaX
     -> PaX control
        -> Use ELF program header Marking
     -> Non executable pages
        -> enforce non-executable pages
        -> segmentation based non-exec pages
        -> enforce non-executable kernel pages
     -> address space layout randomization
        -> Randomize kernel stack base
```

Et ce qui vous semblera pertinent selon votre configuration.

Enfin :

```
make bzImage && make install
```

ou si vous ne désirez pas utiliser l'installeur standard:

```
cp arch/i386/boot/bzImage /boot
```

et modifiez ou relancez votre bootloader.

4. Compilation d'OpenVPN

Télécharger l'archive sur openvpn.sourceforge.net : http://prdownloads.sourceforge.net/openvpn/openvpn-2.0_rc6.tar.gz

Pour OpenVPN, rien de particulier si ce n'est d'activer pthreads pour de meilleurs résultats :

```
tar zxvf openvpn-2.0_rc6.tar.gz
cd openvpn-2.0_rc6
./configure --enable-pthread
make
make install
```

testez que la compilation a bien marché et que la machine dispose de tout ce qu'il faut pour héberger des VPN :

dans un shell, lancer :

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

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

Dans un autre shell, lancer un client :

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

Ca devrait tourner quelques minutes (environ deux) et se finir sans soucis, sinon c'est que l'une des étapes précédentes n'a pas fonctionné ou été effectuée.

5. Fichiers de démarrage

Un fichier d'exemple de configuration est disponible dans le répertoire sample-config-files, mais j'ai préféré faire le mien, à adapter (il marche pour une debian) :

A stocker dans :

/etc/init.d/openvpn

```
#!/bin/bash

DAEMON=/usr/local/openvpn/openvpn
CONF=/usr/local/openvpn/etc/openvpn.conf
CONFIG_DIR=/usr/local/openvpn/etc
DAEMONARG="--daemon openvpn"
PIDFILE=/var/run/openvpn.pid

test -x $DAEMON || exit 0
test -d $CONFIG_DIR || exit 0

case "$1" in
start)
  echo -n "Starting openvpn : "
  $DAEMON --writepid $PIDFILE --config $CONF $DAEMONARG \
         --cd $CONFIG_DIR --chroot /usr/local/openvpn
  echo "Done."
```

```
;;

stop)
  echo -n "Stopping openvpn : "
  PID=`cat $PIDFILE`
  kill $PID
  rm $PIDFILE
  echo "Done."
;;

restart)
  echo "Restarting openvpn : "
  echo ""
   sh $0 stop
   sh $0 start
  echo ""
  echo "Openvpn has restarted."
;;

*)
  echo "Usage: $0 {start|stop|restart}" >&2
  exit 1
  ;;
esac

exit 0
```

Pour un démarrage automatique d'OpenVPN à chaque reboot (sous debian) taper :

```
chmod 755 /etc/init.d/openvpn
ln -s /etc/init.d/openvpn /etc/rc2.d/S90openvpn
ln -s /etc/init.d/openvpn /etc/rc6.d/K90openvpn
```

Si vous souhaitez utiliser un serveur OpenVPN et bridger votre interface locale, un script de bridging comme celui-ci par exemple peut vous être utile. (il vous faut avoir installé brctl. Ce script ne vaut que si vous regrouper tous vos clients sur un même serveur, si vous utilisez plusieurs devices tun ou tap, préférez le script livré dans le répertoire sample-scripts d'openvpn). N'oubliez pas de remplacer les noms des interfaces par les votres. Dans mon cas eth1 est l'interface de LAN que je veux bridger avec l'interface VPN.

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

```
#!/bin/bash

brctl addbr br0
brctl addif br0 eth1
brctl addif br0 tap0
ifconfig tap0 0.0.0.0 promisc up
ifconfig eth1 0.0.0.0 promisc up
ifconfig br0 192.168.0.254 netmask 255.255.255.0 broadcast 192.168.0.255
```

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

```
#!/bin/bash

ifconfig br0 down
brctl delbr br0
ifconfig tap0 10.0.0.1 netmask 255.255.255.0 broadcast 10.0.0.255
ifconfig eth1 192.168.0.254 netmask 255.255.255.0 broadcast 192.168.0.255
```

Penser a mettre un chmod 755 sur /etc/init.d/bridge-st* Ne lancer le bridge qu'après avoir lancer OpenVPN sinon les interfaces à bridger ne seront pas encore présentes. Si vous souhaitez le démarrer automatiquement, c'est même le principe qu'avec le fichier /etc/init.d/openvpn. Encore une fois, n'oubliez pas les chmod 755 qui vont bien.

6. Génération des clefs

Le système de génération des clefs et des certificats a été simplifié par l'auteur d'OpenVPN pour que cela ne devienne par une corvée. Le répertoire easy-rsa dans le répertoire des sources contient les scripts nécessaires à une configuration rapide des clients et du serveur.

Globalement il faut générer plusieurs type de clefs et certificats :

- une clef de pour le protocole d'échange diffie-hellman (dh1024.key)
- une clef et un certificat pour l'autorité de certification (a moins que vous en utilisiez une officielle comme par exemple www.verisign.com ou www.freessl.com, mais cela revient très cher). (ca.crt, ca.key)
- une clef et un certificat pour le serveur (vpnserver.key, vpnserver.crt)
- une clef et un certificat pour chaque client ([nom client].crt, [nom client].key
- une clef qui sert à autoriser les accès au démon (ta.key)

ensuite tapez :

```
cd easy-rsa
```

Editez le fichier vars et régler le de façon correcte. A titre d'exemple, voici ce que le mien contient :

```
export D=/usr/src/openvpn/easy-rsa
export KEY_CONFIG=$D/openssl.cnf
export KEY_DIR=$D/keys
export KEY_SIZE=1024
export KEY_COUNTRY=FR
export KEY_PROVINCE=NA
export KEY_CITY=PARIS
export KEY_ORG="Serveur VPN"
export KEY_EMAIL="philippe.humeau@nbs-system.com"
```

(ceci va charger les précédentes variables en mémoire pour que les scripts puissent les utiliser)

Faite :

```
source ./vars
mkdir keys
touch keys/index.txt
echo 01 > keys/serial
```

Enfin, générer les différentes clefs :

```
./build-dh
./build-ca
./build-key-server [NOM DE VOTRE SERVEUR]
```

Vous pouvez aussi dès a présent générer les clefs des clients :

```
./build-key [nom de votre client]
```

N'oubliez pas plusieurs points importants : les fichiers .key sont " top secret ", ils ne doivent en aucun cas être donnés ou diffusés. Si vous devez les transférer à vos clients, faite le par des voies sécurisées. Toute (enfin une bonne part) votre sécurité du VPN repose sur ce point.

Enfin, il vous faut générer une clef spéciale pour éviter les attaques dites " Man in the middle " ou " homme du milieu " en français. (ce sont des attaques d'interception de clefs).

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

Faite :

```
mv keys/ ..
chown -R nobody.nobody /usr/local/openvpn
```

Les différentes clefs et certificats sont maintenant stockées dans le répertoire keys/
Voila, vous avez générer tout ce qu'il faut, on peut passer à la configuration des clients et du serveur.

NOTE : N'oubliez pas d'utiliser des CN (Common Name) unique pour chaque participant du VPN (serveur, et chaque client), sinon votre serveur ne fonctionnera pas ! Mettez aussi un ON (Organization Name) Commun au serveur et aux clients.

7. Configuration du Serveur pour le mode Ethernet Bridge

Vous avez peut être installé votre serveur dans un autre répertoire (directive -prefix pendant le ./configure) mais sinon tout doit se trouver dans /usr/local/openvpn.

Personnellement, j'ai créé un sous répertoire etc/ dans /usr/local/openvpn. Etant donné que je chroot mes daemons, c'est un réflexe. Utiliser chroot est une bonne idée car cela permet (encore plus combiné à GRsec) de protéger votre serveur contre un nombre important d'attaques. Je stock donc ma configuration d'OpenVPN server dans /usr/local/openvpn/etc. Les scripts de démarrage du début de ce howto sont d'ailleurs fait selon ce principe.

Pour un serveur bridgé, voici ma configuration actuelle :
Fichier /usr/local/openvpn/etc/openvpn.conf :

```
local [IP a laquelle vos clients se connecteront]
port [PORT sur lequel vos clients se connecteront]
proto udp
dev tap
mode server
tls-server
tun-mtu 1500
mssfix
persist-key
persist-tun
ca ../keys/ca.crt
cert ../keys/server-vpn.crt
key ../keys/server-vpn.key  # This file should be kept secret
dh ../keys/dh1024.pem
tls-auth ../keys/ta.key 0# This file is secret too
server-bridge 192.168.0.253 255.255.255.0 192.168.0.5 192.168.0.15
ifconfig-pool-persist /usr/local/openvpn/log/ipp.txt
client-to-client
keepalive 10 120
```

```
cipher BF-CBC
comp-lzo
max-clients 15
user nobody
group nobody
chroot /usr/local/openvpn
status /usr/local/openvpn/log/status.log
log-append /usr/local/openvpn/log/openvpn.log
verb 4
mute 10
```

Comme le montre ce fichier de configuration, j'ai stocké mes log dans /usr/local/openvpn/log et mes clefs dans /usr/local/openvpn/keys N'oubliez pas de créer le device TUN dans le /dev :

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

Directives :

Local
l'IP publique de votre machine. (IP Internet)

Port
Le port sur lequel le serveur attend les connexions. Par défaut il est fixé à 1194. Je vous conseil de changer ce port pour éviter que des personnes mal intentionnées n'identifie trop facilement ce port comme étant un OpenVPN au cas ou le daemon connaitrait un jour une faille de sécurité. Ce n'est pas une grosse protection, mais en sécurité comme ailleurs, les petits cour d'eau font les grandes rivières. (En gros c'est toujours ça de pris à l'ennemi)

dev tap
signifie que vous allez utiliser le mode ethernet bridging, via une interface tap

ca
Le chemin ou vous avez stocké le fichier ca.crt (certificat de l'autorité de certification)

cert
le chemin où vous avez stocké le certificat du serveur

key
le chemin où vous avez stocké la clef privée du serveur

dh
le chemin où vous avez stocké la clef diffie-hellman

tls-auth
le chemin où vous avez stocké la clef anti-M2M (et 0 est le type de connexion, ici serveur)

server-bridge
Une directive liée aussi au bridging. En fait chaque client se connectant obtiendra, ici, une IP entre 192.168.0.5 et 192.168.0.15 par une méthode DHCP.

ifconfig-pool-persist
définit le fichier qui stock les correspondences entre ip dynamique et nom d'host. Il permet de ré-attribuer les mêmes IP aux mêmes clients en fonction du champ Common Name de leur certificat. Si vous souhaitez attribuer certaines IP à certains client, éditer le en ce sens. Par exemple :

```
[CN client 1],192.168.0.6
[CN client 2],192.168.0.7
[CN client 3],192.168.0.8
[CN client 4],192.168.0.9
```

client-to-client
permet aux différents clients de se voir entre eux. (et pas juste voir le serveur)

cipher BF-CBC
définie le type d'encryption. Ici le Blow Fish est sélectionné. Il est à la fois suffisamment sécurisé et rapide pour avoir de bonnes performances.

Directives supplémentaires :

Il existe de nombreuses directives, mais toutes ne sont pas décrites ici. On peut aussi citer fragment 1300 (qui demande explicitement à fragmenter tout paquet supérieur à 1300 octets). Mssfix, Fragment et les tailles des MTU peuvent beaucoup influer sur les performances du VPN, régler les en fonction de vos besoins.

8. Installation du client

Il vous faut télécharger :

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

Il existe un moyen de compiler son propre client (éventuellement pour déployer directement des fichiers de configurations pré édités et les clefs de connexions).

Vous devez ensuite mettre un fichier openvpn.conf, les fichiers ta.key, ca.crt, [clef client].key et [clef client].crt (que vous avez générer auparavant avec le script build-key) dans le répertoire config. (Généralement c:\program files\openvpn\config)

9. Configuration du Client pour le mode Ethernet Bridge

En fonction de la configuration serveur vue au chapitre " Configuration du Serveur pour le mode Ethernet Bridge ", voici un fichier client adapté :

Fichier openvpn.conf

```
client
```

```
dev tap
proto udp
remote [IP PUBLIQUE DU SERVEUR] [PORT DU SERVEUR]
resolv-retry infinite
nobind
tls-client
persist-key
persist-tun
ca ca.crt
cert client1.crt
key client1.key
ns-cert-type server
tls-auth ta.key 1
cipher BF-CBC
comp-lzo
verb 2
mute 5
```

Le client à donc besoin de son fichier d'installation mais aussi des fichiers ta.key, du certificat de la Certification Authority (CA) (ca.crt), de sa clef privé et de son certificat.

Il faut les copier dans c:\program files\openvpn\config si vous avez tout compilé et installé par défaut. Remplacer [IP PUBLIQUE DU SERVEUR] et [PORT DU SERVEUR] par l'IP du serveur VPN et le port de connexion choisi (1194 par défaut).

10. Configuration du Serveur pour le mode Routed

Fichier openvpn.conf

```
local [VOTRE IP PUBLIQUE]
port [VOTRE PORT]
proto udp
dev tun
mode server
tls-server
tun-mtu 1500
mssfix
persist-key
persist-tun
ca ../keys/ca.crt
cert ../keys/server-vpn.crt
key ../keys/server-vpn.key  # This file should be kept secret
dh ../keys/dh1024.pem
tls-auth ../keys/ta.key 0 # This file is secret
ifconfig 192.168.0.1 192.168.0.2
route 192.168.0.0 255.255.255.0
push "route 192.168.0.0 255.255.255.0"
client-config-dir ccd
ccd-exclusive
client-to-client
keepalive 10 120
cipher BF-CBC
comp-lzo
max-clients 15
user nobody
group nobody
chroot /usr/local/openvpn
status /usr/local/openvpn/log/status.log
log-append /usr/local/openvpn/log/openvpn.log
verb 4
mute 10
```

La plupart des paramètres restes les mêmes, mais certaines nuances assez subtiles sont importantes :

dev tun
définit que nous travaillerons en mode routé et non bridgé.

ifconfig
La directive ifconfig 192.168.0.1 192.168.0.2 configure l'interface tun du serveur

route
La directive route 192.168.0.0 255.255.255.0ajoute une route statique au serveur VPN pour joindre les clients du VPN. La directive push "route 192.168.0.0 255.255.255.0" envoi la route par défaut au client lors de sa connexion.

client-config-dir
J'utilise aussi (mais ce n'est pas obligatoire) " client-config-dir ccd " qui permet de forcer les clients à utiliser les paramètres contenu dans le fichier ccd/[CN Client].

Si vous ne souhaitez pas utiliser le mode " ccd " commentez les directives ccd-exclusive et client-config-dir.

Par exemple si un client à un common name (CN) de type laptop-pierre, un fichier dans le répertoire /usr/local/openvpn/ccd/laptop-pierre contenant :

```
ifconfig-push 192.168.0.14 192.168.0.13
iroute 192.168.0.0 255.255.255.255
```

donnera au laptop de pierre une IP de 192.168.0.14, une passerelle vers le VPN de 192.168.0.13 et une route par défaut pour le réseau 192.168.0.0/24.

L'intérêt est de pouvoir contrôler l'activité des clients et les filtrer. De cette façon on est sûr de l'IP qui leur est attribué et du routage. De la même façon on peut " pousser " plusieurs routes différentes au client.

ccd-exclusive

La directive ccd-exclusive implique que si le client n'a pas de fichier dans le répertoire /usr/local/openvpn/ccd pour lui, il ne pourra pas se connecter.

NOTE IMPORTANTE : point critique de la configuration en mode routé :

Le client en mode routé obtient une IP, un réseau, un broadcast et un gateway. Pour que cela fonctionne dans tous les cas, sous Windows, le driver de carte réseau TAP-WIN32 a besoin de travailler avec un sous réseau de 252.

Votre machine cliente aura donc par exemple :
IP: 192.168.0.14
NETMASK: 255.255.255.252
GW vers VPN: 192.168.0.13
Réseau: 192.168.0.12
Broadcast: 192.168.0.15

Ensuite c'est le daemon serveur OpenVPN qui s'occuper de relier ces masques de sous réseau ensemble si vous avez choisi le mode client-to-client.

Ainsi vous ne pouvez pas adresser n'importe comment vos clients, et toutes les IP ne sont pas disponibles. Les couples d'IP (IP du client, IP du serveur pour le client) sont donc limités aux possibilités suivantes (openvpn -show-valid-subnets pour les retrouver) :

[1, 2] [5, 6] [9, 10] [13, 14] [17, 18] [21, 22] [25, 26] [29, 30] [33, 34] [37, 38] [41, 42] [45, 46] [49, 50] [53, 54] [57, 58] [61, 62] [65, 66] [69, 70] [73, 74] [77, 78] [81, 82] [85, 86] [89, 90] [93, 94] [97, 98] [101,102] [105,106] [109,110] [113,114] [117,118] [121,122] [125,126] [129,130] [133,134] [137,138] [141,142] [145,146] [149,150] [153,154] [157,158] [161,162] [165,166] [169,170] [173,174] [177,178] [181,182] [185,186] [189,190] [193,194] [197,198] [201,202] [205,206] [209,210] [213,214] [217,218] [221,222] [225,226] [229,230] [233,234] [237,238] [241,242] [245,246] [249,250] [253,254]

Et n'oubliez pas que dans l'exemple nous avons attribué 1 et 2 au serveur, donc le premier client aura l'IP 192.168.0.6.

11. Configuration du Client pour le mode Routed

```
client
dev tun
proto udp
remote [IP PUBLIQUE DU SERVEUR] [PORT DU SERVEUR]
resolv-retry infinite
nobind
tls-client
persist-key
persist-tun
ca ca.crt
cert client1.crt
key client1.key
ns-cert-type server
tls-auth ta.key 1
cipher BF-CBC
pull
comp-lzo
verb 2
mute 5
```

pull

La directive pull (tirer) va obliger le client à demander au serveur de lui pusher (pousser) la configuration qu'il a pour lui. Et comme dans mon exemple nous utilisons le ccd (client configuration directory), il va prendre ses informations depuis le fichier /usr/local/openvpn/ccd/client1

12. Firewall

L'objet de ce HOWTO n'est pas de vous former au firewalling, mais dans le principe, il est assez complexe de filtrer un VPN qui a justement pour but de rendre perméable deux interfaces réseaux, voir deux réseaux (en mode bridging). Toujours est-il que je vous recommande fortement de filtrer vos clients. Dans un premier temps, les règles suivantes devraient vous permettre de faire marcher votre VPN, ensuite à vous de restreindre toute cela de la façon adéquate pour votre réseau.

N'oubliez pas aussi que si vous bridgez vos interfaces sur le serveur de VPN, vous devez filtrer sur br0 et non plus sur eth0 ou eth1. (il est possible aussi de filtrer sur tap0 avec des -s et des -d en iptables)

Jeu de règles (ultra permissif) pour faire marcher votre VPN :

```
# pour accepter les connexions daemon OpenVPN
iptables -I INPUT -p udp --dport [NUMERO DU PORT DU SERVEUR] -j ACCEPT
```

```
# pour les tunnels VPN routés
```

```
iptables -I INPUT -i tun0 -j ACCEPT
iptables -I FORWARD -i tun0 -j ACCEPT
iptables -I FORWARD -o tun0 -j ACCEPT
iptables -I OUTPUT -o tun0 -j ACCEPT
```

```
# Pour les VPN bridgés
iptables -I INPUT -i tap0 -j ACCEPT
iptables -I FORWARD -i tap0 -j ACCEPT
iptables -I FORWARD -o tap0 -j ACCEPT
iptables -I OUTPUT -o tap0 -j ACCEPT
iptables -I INPUT -i br0 -j ACCEPT
iptables -I FORWARD -i br0 -j ACCEPT
iptables -I OUTPUT -o br0 -j ACCEPT
```

Ces règles autorisent les paquets à transiter sur les device TUN ou TAP. N'utiliser que les règles adaptées en fonctions de vos devices, tun en routé et tap en bridgé. Si vous lancez plusieurs instances du daemon, pensez à filter aussi tun1,tun 2... ou tap 1,2,3 etc... ou plus simple fait tap+ ou tun+ à la place (le + s'occupe de toutes les interfaces du type décrit. Tun+, toutes les interfaces tun)

Les règles sont écrites avec un paramètre -I (insert) au lieu de -A (append) pour ne pas avoir de perturbation de la part d'autres règles du firewall. Pensez à les remettre en -A une fois vos tests effectués pour que les autres règles du firewall ne soient pas ignorées.

Faites attention aux règles du style :

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

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

13. Les petits trucs

Pour simplifier la vie des mes utilisateurs, je parse le fichier status.log et je publie les résultats sur une page HTML sur l'interface du serveur VPN avec un apache. Ainsi chacun peut savoir qui est en ligne, depuis combien de temps et à qui appartient une IP particulière. Un script cron lance régulièrement la mise à jour de la page web.

entrée dans /etc/crontab :

```
1-60/5  *    * * *   root    /usr/local/openvpn/scripts/status.sh
```

Je vous conseille pour cela de mettre les entrées de vos utilisateurs dans le fichier /etc/hosts de votre linux. De la même façon mettez un fichier hosts dans les répertoires c:\windows\system32\drivers\etc. Ce n'est pas indispensable mais c'est plus confortable.

Script de génération de la page web d'état des connexions : (marche pour un serveur bridgé, il faut l'adapter pour un routé)

Fichier /sur/local/openvpn/scripts/status.sh

```
#!/bin/bash

LOG=/usr/local/openvpn/log/status.log
STATUS=/var/www/status.html

echo "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">" > ${STATUS}
echo "<HTML>" >> ${STATUS}
echo "<HEAD>" >> ${STATUS}
echo "  <TITLE>Speedball</TITLE>" >> ${STATUS}
echo "</HEAD>" >> ${STATUS}
echo "<meta http-equiv="refresh" content="60">" >> ${STATUS}
echo "<body >" >> ${STATUS}
echo """>> ${STATUS}
echo "<H1>VPN Status</H1>" >> ${STATUS}
echo """>> ${STATUS}
echo """>> ${STATUS}

nbhosts=`cat ${LOG} | grep 00:ff | wc -l`


echo "<p class="howto">There are $nbhosts online hosts :</P>" >> ${STATUS}
echo """>> ${STATUS}

hosts_ether=`cat ${LOG} | grep 00:ff | cut -f 1 -d,`
hosts_name=`cat ${LOG} | grep 00:ff | cut -f 2 -d,`
update=`cat ${LOG} | grep Updated`

for a in $hosts_name
do
  pingtime=`ping -c 3 -q $a | grep avg | cut -f 2 -d = | cut -f 2 -d/ | cut -f 1 -d.`
  vpnip=`ping -c 1 -q $a | grep "data" | cut -f 3 -d " "`
  pubip=`cat ${LOG} | grep -v 00:ff | grep $a | cut -f 2 -d, | cut -f 1 -d:`
```

```
  connexion_time=`cat ${LOG} | grep -v 00:ff | grep $a | cut -f 5 -d,`

  echo -n "<p class="howto">$a      IP: $vpnip - $pubip, \
Ping : $pingtime ms, Connected since : $connexion_time</P>" \
>> ${STATUS}
done

echo """>> ${STATUS}
echo "<p>Data updated on : `echo $update | cut -f 2 -d,`.</p>"> ${STATUS}
```

Bon ça vaut ce que ça vaut, c'est-à-dire pas grand-chose, mais c'est fait à 3H du mat' un peu fatigué. Ceci dit ça marche je pense, j'avais un problème qui vient du fait que la table arp ne garde pas son entrée à une machine qui ne s'est pas signalée depuis un moment. Ce qui m'oblige à la pinger pour rafraîchir son entrée dans la table arp et ainsi obtenir son ip VPN. Ca fonctionne du moment que vous avez renseigné correctement le fichier /etc/hosts, mais bon il y a sûrement mieux à faire, mais en un quart d'heure, je n'ai pas trouvé.

14. Considération de sécurité

Encore une fois, le VPN peut poser des problèmes de sécurité non négligeables. La sécurité ne vaut que par son maillon le plus faible et si l'un de vos clients se fait voler sa clef privée, vous risquez de compromettre les machines du VPN et le serveur. Chaque machine doit donc être correctement configurée, dotée d'un firewall personnel, d'un anti virus, et si possible d'un IPS et d'un anti spyware.

Un virus du type Blaster sur un portable qui entre dans le VPN et tout le monde se retrouve avec Blaster par exemple.

Il est donc conseillé de bien protéger les membres du VPN et de les filtrer sur le firewall. L'utilisation du daemon OpenVPN en lui-même, surtout s'il est chrooté et protégé par GRsec n'est pas dangereuse, mais le danger vient plutôt des clients en général.

15. Lancement des connexions & Tests

Pour tester votre VPN, lancer la connexion sur un client (sous Windows, un double clique sur l'icône dans la barre de tache suffit). Une fenêtre de log apparaît et couplée avec un tail -f /usr/local/openvpn/log/openvpn.log sur le serveur, rien ne devrait être réellement problématique. Ensuite, quelques pings dans un sens, dans l'autre et entre clients devrait vous confirmer que tout marche correctement.

Ben non, ca marche pas !

Il existe beaucoup de raison pour lesquelles cette installation pourrait ne pas marcher, je vais en citer et expliquer quelques unes, au cas où la votre serait dans le lot, vous pourrez gagner du temps.

Sous Windows 2000 : des problèmes de routages peuvent survenir, il arrive que l'interface VPN ne récupère pas les routes nécessaires. Vous pouvez alors taper la commande suivante dans un shell (cmd.exe) pour vous en assurez : route print

Si la route du VPN n'y est pas, faite un : route add [adresse du réseau VPN] MASK 255.255.255.0 [IP du serveur VPN]

Firewall sur le serveur : Si vous ne recevez pas les connexions des utilisateurs, il est probable que vous n'ayez pas ouvert (ou forwardé) le port dans votre firewall. Par défaut c'est le 1194 en udp, ce qui donne en iptables :

```
iptables -I INPUT -s 0.0.0.0/0 -d [IP PUB DU SERVEUR] -p udp -dport 1194 -j ACCEPT
```

Firewalls personnels : Vos client ont peut être des firewalls personnels sur leurs machines. Le SP2 de XP ne devrait pas poser de problèmes dans son réglage par défaut, mais les firewalls personnels doivent au moins être configurés pour accepter les paquets du serveur et éventuellement du réseau VPN si tel est votre désir.

Certificats " self signed " : Typiquement vous avez généré des certificats ou le champ CN du client est le même que celui du serveur. Régénérez vos certificats avec des CN (Common Name) différents pour chaque entité du VPN.

Paramètres réseau complètement bidon : vous avez probablement mis les clients en " dev tun " et le serveur en " dev tap " ou l'inverse. Mettez dev tun ou dev tap dans le fichier de configuration du client et du serveur.

16. Statistiques Personnelles

Avec machine de type K6 II 266 Mhz et 160 Mo de RAM, j'obtiens les résultats suivants :

% CPU utilisé : 6% en moyenne par client actif
% BP pour le VPN : environ 10% de plus de trafic par rapport à une connexion normale.
Ping : identique au ping sans VPN, sauf en cas de saturation de la bande passante montante de ma machine ou là les temps de ping sont multipliés par un facteur allant jusqu'à 4.

Donc en résumé, mon K6 II 266 Mhz et ma ligne actuelle (6 Mb/s en DL et 512 Kb/s en UL) me permettent d'héberger environ 8 utilisateurs qui consomment peu de BP. (Car c'est votre bande passante en upload qui va limiter la vitesse des échanges entres les clients VPN)

17. Conclusion

J'ai peut être fait un ou deux oublis dans la configuration du serveur en mode route car je l'ai refaite de tête, mais celà devrait fonctionner, au pire il faudra modifier ou ajouter une directive du fichier de configuration. N'hésitez pas à me le signaler afin que je corrige ce HOWTO.

En conclusion, OpenVPN est une alternative souple, robuste et efficace aux serveurs VPN IPSEC souvent très coûteux et pas toujours simple à mettre en place, surtout dans les cas de NAT. De l'OpenSource pur souche au service des administrateurs et des utilisateurs !

Un grand bravo à James Yonan qui a fait (et qui fait) progresser ce domaine à pas de géant.

18. Webographie

http://www.nbs-system.com
http://openvpn.sourceforge.net
ftp://ftp.kernel.org/pub/linux/kernel
http://prdownloads.sourceforge.net/openvpn/openvpn-2.0_rc6.tar.gz
http://prdownloads.sourceforge.net/openvpn/openvpn-2.0_beta6-install.exe
http://www.ntkernel.com/utilities/etherbridge.shtml

http://www.oberhumer.com/opensource/lzo
http://www.openssl.org
http://www.grsecurity.net
http://www.gnu.org/copyleft/gpl.html
http://www.netfilter.org/documentation/HOWTO/fr/netfilter-hacking-HOWTO.html

Autres HOWTO sur OpenVPN (en anglais)

http://www.pavelec.net/adam/openvpn/bridge
http://www.nwfusion.com/news/2004/122004cheap/securite/rpv-vpn-1.html
http://www.linuxjournal.com/article/7949
http://fedoranews.org/contributors/florin_andrei/openvpn/
http://www.osnews.com/story.php?news_id=5803
http://www.linuxsecurity.com/feature_stories/feature_story-152.html
http://www.sans.org/rr/whitepapers/vpns/1459.php
http://www.oreillynet.com/pub/a/security/2004/10/21/vpns_and_pki.html
http://www.sans.org/rr/papers/20/1459.pdf
http://openvpn.net/papers/openvpn-101.pdf
http://www.nilings.se/openvpn/files/howto/openvpn-howto_roll_your_own_installation_package.html
http://mia.ece.uic.edu/~papers/volans/settingupCA.html
http://slackerbit.ch/archives/2002/12/11/securing_wifi_with_open/securite/rpv-vpn-1.html
http://www.shorewall.net/OPENVPN.html
http://chrisp.de/en/rsrc/open/securite/rpv-vpn-1.html

Et plus généralement : http://openvpn.net/articles.html

(up)