

Configuration d'une connexion VPN sur un client Debian Linux

<http://lifc.univ-fcomte.fr/prj/lifc/>

Historique des versions	
\$Revision: 003 \$	\$Date: 2009-09-09 08:40:00 +0100 \$
Mise en ligne du document.	
Résumé	
Ce document décrit la procédure de création d'une connexion VPN sur un système Debian Linux ou dérivé (Ubuntu, Kubuntu, Knoppix, etc).	

Table des matières

1. Introduction aux Réseaux Privés Virtuels	1
2. Configuration d'un client Linux pour les initiés	2
2.1. Connexion IPSec	2
2.2. Connexion L2TP	3
2.3. Script pour lancer ou stopper une connexion VPN	5

1. Introduction aux Réseaux Privés Virtuels

L'objectif d'un VPN est simple. Il s'agit de sécuriser des échanges de données en utilisant comme support un réseau non sécurisé comme par exemple le réseau Internet. Un VPN est également un bon candidat pour la sécurisation de flux informatiques à travers un réseau WIFI.

Les concepts sous-jacents :

- Un réseau informatique : c'est un ensemble de technologies matérielles et logicielles permettant de faire circuler des flux informatiques sur un support : câbles, fibres optiques, ondes radio, courant porteur, etc

Par défaut, les flux de données ne sont pas sécurisés. Dans le cas d'un réseau local, on « accepte » en général que les données circulent « en clair » car les utilisateurs font partie de l'entreprise. La confiance est donc de mise.

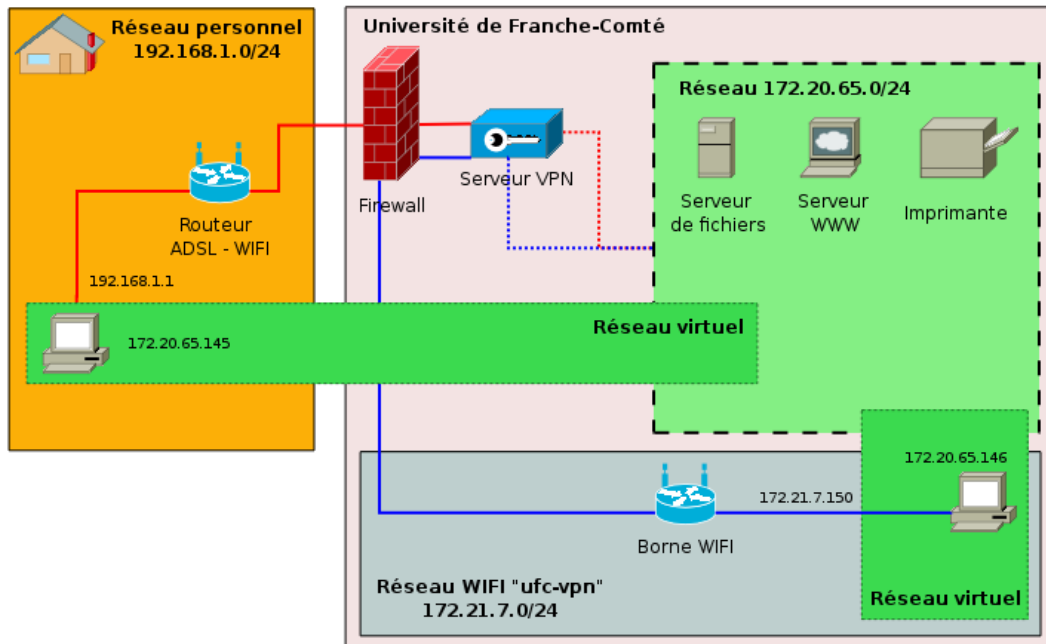
- Un réseau sécurisé (ou privé) : les données circulant sur ce type de réseau sont cryptées. De cette façon, des personnes mal intentionnées pourront toujours « écouter » les données qui transitent mais ne pourront pas en connaître le contenu.
- Un réseau virtuel : un réseau virtuel est un réseau qui n'existe pas physiquement. Ce réseau utilise donc pour faire transiter ces données le support de réseaux physiques existants.

Un réseau privé virtuel est donc l'association de ces trois concepts. Une fois en place, il offre la possibilité d'utiliser un réseau public non sécurisé pour créer un réseau privé (données cryptées) et y faire circuler des données.

Un VPN permet de passer par une connexion Internet de type ADSL pour accéder à un réseau interne de manière transparente. Un fois connecté, le poste nomade peut se servir des ressources (espaces disques, imprimantes, etc) fournies par ce réseau.

Dans le schéma ci-dessous, nous observons une connexion établie entre un ordinateur personnel (accès à Internet de type ADSL) et le réseau universitaire. Les traits rouges montrent les liens physiques réels. La zone « Réseau virtuel » montre le VPN mis en place entre son poste et le réseau interne après l'établissement de la connexion.

Une autre connexion est établie entre un poste situé dans le réseau WIFI "ufc-vpn" et le réseau universitaire. Le réseau WIFI "ufc-vpn" permet de créer un réseau virtuel entre ce poste nomade et un réseau interne. Tout comme le premier exemple, le poste fait partie du réseau interne et peut accéder à toutes les ressources.



2. Configuration d'un client Linux pour les initiés

Cette section décrit une connexion IPSec/L2TP en utilisant les outils en ligne de commande sous un système Linux.

2.1. Connexion IPSec

Pour pouvoir créer un tunnel IPSec, nous devons installer le paquet `openswan` :

```
# apt-get install openswan
```

Nous devons également posséder le certificat X509 et la clé privée associée plus le certificat de l'autorité de certification. Dans l'exemple, nous utiliserons les noms de fichiers :

- `demovpn_demolinux-cert.pem` pour le certificat
- `demovpn_demolinux-key.pem` pour la clé privée
- `ca-vpn.pem` pour le certificat de l'AC (Autorité de Certification)

Nous effectuons les actions suivantes :

1. nous copions le fichier `demovpn_demolinux-cert.pem` dans le répertoire `/etc/ipsec.d/certs`
2. nous copions le fichier `demovpn_demolinux-key.pem` dans le répertoire `/etc/ipsec.d/private`
3. nous copions le fichier `ca-vpn.pem` dans le répertoire `/etc/ipsec.d/cacerts`

Nous ajoutons au fichier `/etc/ipsec.secrets` :

```
: RSA /etc/ipsec.d/private/demovpn_demolinux-key.pem "motdepasse"
```

Le « motdepasse » nous a été demandé lors de la création du couple clé/certificat. A défaut, il nous est communiqué par la personne ayant créé le certificat.

Nous supprimons le contenu du fichier `/etc/ipsec.conf` et nous ajoutons :

```
version 2.0

config setup
    nat_traversal=yes
    uniqueids=yes
    nhelpers=0
```

```
conn ufc-vpn
    left=%defaultroute
    leftnexthop=192.168.1.254
    leftsendcert=always
    leftcert=demovpn_demolinux-cert.pem
    leftrsasigkey=%cert
    right=194.57.91.250
    rightca=%same
    rightrsasigkey=%cert
    keyingtries=3
    compress=no
    authby=rsasig
    type=tunnel
    auto=add
    pfs=no

include /etc/ipsec.d/examples/no_oe.conf
```

Nous devons également apporter quelques modifications au paramétrage du noyau. Voici le contenu d'un fichier `/etc/sysctl.conf` fonctionnel :

```
net.ipv4.ip_forward=1
net.ipv4.icmp_ignore_bogus_error_responses=1
net.ipv4.conf.all.log_martians=0
net.ipv4.conf.default.rp_filter=0
net.ipv4.conf.default.accept_redirects=0
net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.all.arp_ignore=1
net.ipv4.conf.all.arp_announce=2
```

Pour prendre en compte les nouveaux paramètres, nous pouvons redémarrer la machine ou utiliser la commande `sysctl -p`. Nous pouvons ensuite contrôler si tout est correct pour IPsec avec la ligne de commande ci-après :

```
# ipsec verify
Checking your system to see if IPsec got installed and started correctly:
Version check and ipsec on-path [OK]
Linux Openswan U2.4.12/K2.6.26-2-686 (netkey)
Checking for IPsec support in kernel [OK]
NETKEY detected, testing for disabled ICMP send_redirects [OK]
NETKEY detected, testing for disabled ICMP accept_redirects [OK]
Checking for RSA private key (/etc/ipsec.secrets) [DISABLED]
  ipsec showhostkey: no default key in "/etc/ipsec.secrets"
Checking that pluto is running [OK]
Checking for 'ip' command [OK]
Checking for 'iptables' command [OK]
Opportunistic Encryption Support [DISABLED]
```

Aucune ligne ne devrait contenir le mot `>FAILED`. Dans le cas contraire, vous devez reprendre le contenu du fichier `sysctl.conf`.

Si nous sommes derrière un routeur ADSL, il faudra en général indiquer à Openswan le chemin à prendre en lui fournissant l'adresse IP du routeur avec l'option `leftnexthop`¹. Dans l'exemple, notre routeur ADSL possède l'adresse IP `192.168.1.254`. Nous redémarrons openswan :

```
# /etc/init.d/ipsec restart
```

Nous montons la connexion IPsec avec la commande :

```
# ipsec auto --up ufc-vpn
```

2.2. Connexion L2TP

Lorsque le tunnel IPsec est monté, nous pouvons passer à la configuration de la connexion L2TP. Il nous faut installer le paquet `xl2tpd`. Le paquet `l2tpd` est à proscrire en tant que client. Régulièrement, il provoque des fautes de segmentations. Sur une distribution basée sur Debian Lenny, nous installons le paquet avec la commande :

¹L'accès à travers le WIFI universitaire nécessite également cette option. Dans ce cas précis, l'adresse IP à fournir sera l'adresse `172.21.7.254`.

```
# apt-get install xl2tpd
```

Nous supprimons le contenu du fichier `/etc/xl2tpd/xl2tpd.conf` et nous y ajoutons les lignes suivantes :

```
[global]
port = 1701
access control = no

[lac ufc]
lns = 194.57.91.250
redial = yes
redial timeout = 5
max redials = 3
length bit = yes
refuse authentication = yes
refuse chap = yes
require pap = yes
name = cri
ppp debug = yes
pppoptfile = /etc/ppp/options.l2tpd.ufc
```

Nous créons le fichier `/etc/ppp/options.l2tpd.ufc` pour y ajouter les lignes :

```
plugin passwordfd.so
user jmcaricand@lifc-lab
defaultroute
replacedefaultroute
noipdefault
usepeerdns
noauth
lcp-echo-interval 20
lcp-echo-failure 10
noaccomp
mtu 1200
mru 1200
lock
```



Nom d'utilisateur

La valeur associée à la clé `user` représente votre login. Notez l'utilisation du suffixe représentant votre réseau de destination. Les étudiants doivent utiliser le suffixe « lifc-edu » pour être placés dans le réseau ENSEIGNEMENT du laboratoire. Les enseignants peuvent utiliser les suffixes « lifc-lab » ou « lifc-edu ». L'utilisation du suffixe « ufc » qui est à proscrire car certains services du laboratoire contrôlent l'origine des requêtes et seront inaccessibles pour les royaumes autres que « lifc-lab » et « lifc-edu ».

Pour établir la connexion L2TP, nous lançons la commande :

```
echo "c ufc passwordfd motdepasse" >/var/run/xl2tpd/l2tp-control
```

Le terme `ufc` utilisé à la connexion correspond au nom du `lac` défini dans le fichier `/etc/xl2tpd/xl2tpd.conf`. Une interface réseau `ppp0` apparaît maintenant dans le résultat de la commande **ifconfig** :

```
# ifconfig
[...]
ppp0      Lien encap:Protocole Point-à-Point
          inet adr:172.20.65.35  P-t-P:172.20.65.2  Masque:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1300  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:3
          RX bytes:66 (66.0 b)  TX bytes:87 (87.0 b)
```



Résolution des problèmes

Si la connexion L2TP ne fonctionne pas, vous pouvez examiner le contenu du fichier `/var/log/syslog`. Ce dernier contient des informations utiles afin d'isoler les problèmes liés à l'échec de la connexion.

Pour stopper la connexion :

```
echo "d ufc" >/var/run/xl2tpd/l2tp-control
```

Lors de la déconnexion, nous devons restaurer l'ancienne route par défaut si il y en avait une. Ce comportement n'est pas prévu d'origine sous Linux. Nous devons mettre en place les outils nécessaires.

Nous créons le script `/etc/ppp/ip-pre-up` :

```
# cat <<EOF > /etc/ppp/ip-pre-up
#!/bin/sh

PATH=/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin
export PATH

PPP_IFACE="$1"
PPP_TTY="$2"
PPP_SPEED="$3"
PPP_LOCAL="$4"
PPP_REMOTE="$5"
PPP_IPPARAM="$6"
export PPP_IFACE PPP_TTY PPP_SPEED PPP_LOCAL PPP_REMOTE PPP_IPPARAM

PPP_TTYNAME=`/usr/bin/basename "$2"`
export PPP_TTYNAME

run-parts /etc/ppp/ip-pre-up.d \
  --arg="$1" --arg="$2" --arg="$3" --arg="$4" --arg="$5" --arg="$6"
EOF

# chmod +x /etc/ppp/ip-pre-up
```

Nous créons un dossier `/etc/ppp/ip-pre-up.d`. Dans ce dossier, nous créons un fichier `save-defaultroute` contenant de quoi mémoriser la route à restaurer :

```
# mkdir /etc/ppp/ip-pre-up.d
# cat <<EOF > /etc/ppp/ip-pre-up.d/save-defaultroute
#!/bin/bash

GW=$(route -n | grep '^0.0.0.0' | tr -s ' ' | cut -d' ' -f2)
[ -n "$GW" ] && echo $GW > /tmp/$PPP_IFACE.defaultroute

exit 0;
EOF

# chmod +x /etc/ppp/ip-pre-up.d/save-defaultroute
```

Dans le dossier `/etc/ppp/ip-down.d`, nous créons le script `restore-defaultroute` :

```
# cat <<EOF > /etc/ppp/ip-down.d/restore-defaultroute
#!/bin/sh -e

[ -f /tmp/$PPP_IFACE.defaultroute ] || exit 0

GW=$(cat /tmp/$PPP_IFACE.defaultroute)
route add default gw $GW

exit 0;

# chmod +x /etc/ppp/ip-down.d/restore-defaultroute
```

2.3. Script pour lancer ou stopper une connexion VPN

Voici un petit script qui permet de lancer ou stopper une connexion VPN :

```
# cat <<EOF > ~/myvpn
#!/bin/bash

case "$1" in
```

```
start)
    ipsec auto --up ufc-vpn
    sleep 2
    echo "c ufc passwordfd motdepasse" >> /var/run/xl2tpd/l2tp-control
    ;;

stop)
    echo "d ufc" > /var/run/xl2tpd/l2tp-control
    sleep 2
    ipsec auto --down ufc-vpn
    ;;

*)
    echo "Usage: $0 {start|stop}"
    ;;
esac
EOF

# chmod +x ~/myvpn
```

Un

```
./myvpn start
```

lancera la connexion alors qu'un

```
./myvpn stop
```

la terminera.