

## OpenVPN primer

There are as many advantages to VPN tunnels as there are different VPN scenarios. One easy implementation is the "OpenVPN via tun-device" solution. An example: you'd like to connect your laptop to your LAN at home so that you can use your mail client without reconfiguring it anytime you switch from home to internet and back. Let's say your mail-server is 192.168.1.10 in your LAN (192.168.1.0/24) at home, and you have got a router/firewall providing access to the Internet. You connect from work or school and want to read mail. OpenVPN can create two virtual devices for you when connecting two computers through an encrypted tunnel. Naturally you then have the possibility of forwarding traffic into the networks behind them, and thus would be "virtually connected" to your LAN behind the firewall. To enable this, either your firewall or a server behind it should run OpenVPN (if you choose a server in your LAN, you'll have to forward the destination port to the OpenVPN server).

Here's what you need to do:

Code Listing 1: Enable the tun module in your kernel: Kernel config - tun module

```
[*] Networking support
    Networking options --->
[] Amateur Radio support --->
IrDA (infrared) subsystem support --->
Bluetooth subsystem support --->
[*] Network device support
    Dummy net driver support
    Bonding driver support
    EQL (serial line load balancing) support
        Universal TUN/TAP device driver support
    // This option must be enabled
```

Make sure this module exists and can be loaded. Next, install OpenVPN and its dependencies.

Code Listing 2: Install OpenVPN

```
emerge openvpn
or
apt-get install openvpn
or use your distribution's install tool.
```

Now on both server and client, create a directory for your configuration:

Code Listing 3: Make directory

```
mkdir /etc/openvpn
mkdir /etc/openvpn/myhomelan
```

Inside that directory, create a shared key for your VPN session and copy that key to the client's directory, /etc/openvpn/myhomelan.

Code Listing 4: Generate shared key

```
cd /etc/openvpn/myhomelan
openvpn --genkey --secret myhomelan-key.txt
```

Now for the tricky part, the routing. It is important that the two tun devices on the client and server use IP addresses from the same subnet. The configuration files shown below list the type of device, the two end-points of the tunnel, the compression method and the UDP-port on which the tunnel is established. Finally privileges are dropped to user and group as listed:

**Code Listing 5: Server-side configuration file /etc/openvpn/myhomelan/local.conf**

```
dev tun                ifconfig 172.16.1.1 172.16.1.20 // IP of the local
// tun device and its peer
secret /etc/openvpn/myhomelan/myhomelan-key.txt
comp-lzo
port 5000
user nobody
group nobody
```

The client's configuration needs the tunnel's destination address. This is often a dynamic DNS address, sometimes a fixed IP, depending on your ISP. You also need to route to your home LAN (192.168.1.0 in our example). You can call a shell script from the configuration file that accordingly sets a route.

**Code Listing 6: Client-side configuration file /etc/openvpn/myhomelan/local.conf**

```
remote // or your VPN
// server's external IP if you have a fixed one
dev tun
ifconfig 172.16.1.20 172.16.1.1 // IP of the local tun
// device and its peer
secret /etc/openvpn/myhomelan/myhomelan-key.txt
comp-lzo
port 5000
user nobody
group nobody
up /etc/openvpn/myhomelan/route.sh // sets up the route
//to the network behind the VPN server
```

The route command would need to set the client's gateway for the network 192.168.1.0 to its peer's address (172.16.1.1 in our setup).

**Code Listing 7: /etc/openvpn/myhomelan/route.sh**

```
#!/bin/bash
route add -net 192.168.1.0 netmask 255.255.255.0 gw 172.16.1.1
```

That's it. Start OpenVPN on the server and the client, and check the devices with ifconfig and the routes with route -n. Success!

From <http://www.gentoo.org/news/en/gwn/20041011-newsletter.xml>