How to set up OpenVPN server with both Linux and Mac OS X clients

(A TechRepublic tip compilation)

October 2010

By Vincent Danen

The three tips in this download were originally published individually in the Linux and Open Source blog and the Macs in Business blog on TechRepublic. Vincent Danen takes you through the steps of setting up OpenVPN server and then how to set up both a Linux client and a Mac OS X client using Shimo.

How to set up an OpenVPN server

By Vincent Danen

Having a virtual private network affords a lot of convenience, particularly for those who want or need to access a remote network from a different location, such as connecting to a work network from home, or vice versa. With the availability of 3G on the road, or wireless hotspots everywhere, being able to connect, securely, to a remote private network from anywhere is ideal.

OpenVPN is one of the most reliable VPN setups around. It's fully open source, it's supported on Linux, Windows, and OS X, it's robust, and it's secure. Unfortunately, configuration can be a bit of a pain, so in a series of upcoming tips, I aim to get you up and running quickly.

To begin, you will need to have OpenVPN installed on the server or system you wish to use as a VPN end-point. Most distributions include OpenVPN; for the server setup, I am using OpenVPN 2.0.9 as provided by the RPMForge repository for CentOS 5.

The first part of this series concentrates on the server, while the second and third parts will concentrate on the configuration of Linux and OS X clients, respectively. So without further ado, let's get our hands dirty.

To begin with, you need to copy some files from the OpenVPN docs directory (typically provided in /usr/share/doc/openvpn-[version]) to create certificates:

```
# cd /usr/share/doc/openvpn-2.0.9
# cp -av easy-rsa /etc/openvpn/
# cd /etc/openvpn/easy-rsa/
# vim vars
```

In the vars file, edit the KEY_* entries at the bottom of the file, such as KEY_COUNTRY, KEY_ORG, KEY_EMAIL, etc. These will be used to build the OpenSSL certificates. Next, it's time to initialize the PKI:

```
# . ./vars
# sh clean-all
# sh build-ca
# sh build-key-server server
```

For the above, and the below client certificates, you can enter pretty much anything for the "Common Name" field, however there is a certain logic to use: "OpenVPN-CA" when generating the Certificate Authority, "server" when generating the server certificate, and "client" or the name of the specific client system for the client certificates. Those certificates are generated with:

```
# sh build-key client1
# sh build-key client2
```

The next step is to generate the Diffie Hellman parameters for the server:

```
# sh build-dh
```

When this is done, you will have a number of files in the keys/ subdirectory. At this point, for the clients, you want to copy the appropriate files to them securely (i.e., via SSH or on a USB stick); the files the clients need are *ca.crt*, *client1.crt*, and *client1.key* (or whatever you named the files when you generated them with the build-key script).

Next, create the OpenVPN server configuration file. To get up and running quickly, copy one of the example config files:

```
# cd /etc/openvpn/
# cp /usr/share/doc/openvpn-2.0.9/sample-config-files/server.conf .
# vim server.conf
```

The aim here is to get this going right away, so we won't examine each of the options in detail. The primary things you want to do are to uncomment the "user" and "group" directives, to make the openvpn process run as the unprivileged "nobody" user. You may also want to change the "local" directive to make it listen to one specific IP address. This would be the IP to which your firewall is forwarding UDP port 1194. As well, you will want to set the "client-to-client" directive to enable it, and also set the "push" directives for route and DNS options. What follows is a comment-stripped server.conf, as an example:

```
local 192.168.10.11
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key  # This file should be kept secret
dh dh1024.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "route 192.168.10.0 255.255.254.0"
push "dhcp-option DNS 192.168.10.12"
push "dhcp-option DOMAIN domain.com"
client-to-client
keepalive 10 120
comp-lzo
user nobody
group nobody
persist-key
persist-tun
status openvpn-status.log
verb 3
```

Finally, copy the required keys and certificates that you previously generated:

```
# cd  /etc/openvpn/
# cp easy-rsa/keys/ca.crt .
# cp easy-rsa/keys/server.{key,crt} .
# cp easy-rsa/keys/dh1024.pem  .
```

And, finally, start the OpenVPN server:

```
# /etc/init.d/openvpn start
```

To get routing set up properly on the server so that remote clients, when they connect, can reach more than just the server itself, you will need to enable IP forwarding. This can be done by the following:

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

You can also do it by editing /etc/sysctl.conf and adding the following (this is a good thing to do as it will ensure that packet-forwarding persists across reboots):

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

You also want to ensure that packets going back to the client system are routed properly. This can be done by changing the route on the gateway of the server's network to route packets to the client network (10.8.0.1/32) through the OpenVPN server (if the server happens to be the gateway as well, you don't have to do anything additional to accomplish this). How this is done largely depends on the operating system of the gateway.

Once this is done, you should be able to ping any machine on the server's LAN from the client, and be able to ping the client from any machine on the server's LAN. For instance, from a machine on the server LAN (not the server):

```
% traceroute 10.8.0.6
traceroute to 10.8.0.6 (10.8.0.6), 64 hops max, 52 byte packets
 1  fw (192.168.10.1)  0.848 ms  0.342 ms  0.249 ms
 2  server (192.168.10.11)  0.214 ms  0.231 ms  0.243 ms
 3  server (192.168.10.11)  0.199 ms !Z  0.443 ms !Z  0.396 ms !Z
% ping 10.8.0.6
PING 10.8.0.6 (10.8.0.6): 56 data bytes
64 bytes from 10.8.0.6: icmp_seq=0 ttl=63 time=17.540 ms
```

And from the client:

```
# traceroute 192.168.10.65
traceroute to 192.168.10.65 (192.168.10.65), 30 hops max, 40 byte packets
 1  10.8.0.1 (10.8.0.1)  22.963 ms  27.311 ms  27.317 ms
 2  10.8.0.1 (10.8.0.1)  27.297 ms !X  27.294 ms !X  27.269 ms !X
# ping 192.168.10.65
PING 192.168.10.65 (192.168.10.65) 56(84) bytes of data.
64 bytes from 192.168.10.65: icmp_seq=1 ttl=62 time=515 ms
```

The setting up of OpenVPN clients will be the subject of two tips in the next week. I've made the assumption that the client is correctly configured here, simply to illustrate how it should look when it all works together, but in the next parts of this series we will get into more depth with the client configuration.

How to set up a Linux OpenVPN client

By Vincent Danen

In a previous tip last week, we looked at setting up an OpenVPN server. Now, I'll take you through the setup of a Linux OpenVPN client. (I have also covered setting up an OS X client on OpenVPN in the Macs in Business blog). The Linux client will be based on CentOS 5 using OpenVPN 2.0.9.

For each client, you will need to have copied the client's certificate and key, as well as the CA certificate, from the server. This should be done in a secure manner so you can ensure the files are not altered in any way, such as using SSH to transfer or a USB stick in your possession. Once they are on the client, copy them to the /etc/openvpn/ directory:

```
# cd /etc/openvpn
# cp ~/client.{key,crt} .
# cp ~/ca.crt .
# cp /usr/share/doc/openvpn-2.0.9/sample-config-files/client.conf .
# vim client.conf
```

In the client.conf, what you need to uncomment are the "user" and "group" directives, to make the openvpn run as the unprivileged" nobody" user rather than root. Also, if your key and certificate files are not named "client.key" and "client.crt" you will need to change the *crt* and *key* directives in the file as well.

An uncommented client configuration file follows, that serves as an example:

```
client
dev tun
proto udp
remote linsec.ath.cx 1194
resolv-retry infinite
nobind
user nobody
group nobody
persist-key
persist-tun
ca ca.crt
cert client1.crt
key  client1.key
comp-lzo
pull dhcp-options
To initiate a startup test, execute:
# openvpn client.conf
Tue Sep 14 17:18:14 2010 OpenVPN 2.0.9 x86_64-redhat-linux-gnu [SSL] [LZO] [EPOLL]
built on Mar  8 2007
...
Tue Sep 14 17:18:15 2010 [server] Peer Connection Initiated with 1.2.3.4:1194
Tue Sep 14 17:18:16 2010 SENT CONTROL [server]: 'PUSH_REQUEST' (status=1)
Tue Sep 14 17:18:16 2010 PUSH: Received control message: 'PUSH_REPLY,route
192.168.10.0 255.255.254.0,route 10.8.0.0,ping 10,ping-restart 120,ifconfig 10.8.0.6
10.8.0.5'
...
Tue Sep 14 17:18:16 2010 /sbin/ip addr add dev tun0 local 10.8.0.6 peer 10.8.0.5
Tue Sep 14 17:18:16 2010 /sbin/ip route add 192.168.10.0/23 via 10.8.0.5
Tue Sep 14 17:18:16 2010 /sbin/ip route add 10.8.0.0/32 via 10.8.0.5
Tue Sep 14 17:18:16 2010 GID set to nobody
Tue Sep 14 17:18:16 2010 UID set to nobody
Tue Sep 14 17:18:16 2010 Initialization Sequence Completed
```

There is a lot more output, but the above includes the important bits. We see here that a connection has been established with the remote server, with the IP address 1.2.3.4. We also see that the routes have been added, for the remote 192.168.10.0/23 network, and the VPN-specific 10.8.0.0/32 network. Now, you can make sure the link is up by using:

```
# ifconfig tun0
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.6  P-t-P:10.8.0.5  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

And finally, make sure it works:

```
# ping 10.8.0.1 -c 2
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=21.1 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=14.8 ms
```

A connection to the remote OpenVPN server has been successfully established. And if you followed the previous tip about setting up the OpenVPN server, you should be able to ping and establish a connection to any other available system on the server's LAN at this point as well.

In order to access machines on the remote network with their FQDN (Fully Qualified Domain Name), you will need to modify /etc/resolv.conf to add a nameserver from the remote network to the top of the list. This can, and probably should, be scripted so that the DNS server is used whenever the VPN connection is established. This will allow you to connect to "server.foo.com" rather than always using IP addresses, like "192.168.10.23". On the server side, ensure that requests from this IP range (10.8.0.0/32) are not being blocked by the DNS server ACLs -- this is an important thing to remember, as it bit me and cost me a day of frustration.

At this point the Linux client is set up. The final part of this series (published in the Macs in Business blog on TechRepublic) shows you how to set up Shimo on OS X to connect to the OpenVPN server and access the remote network's services.

How to set up an OS X OpenVPN client

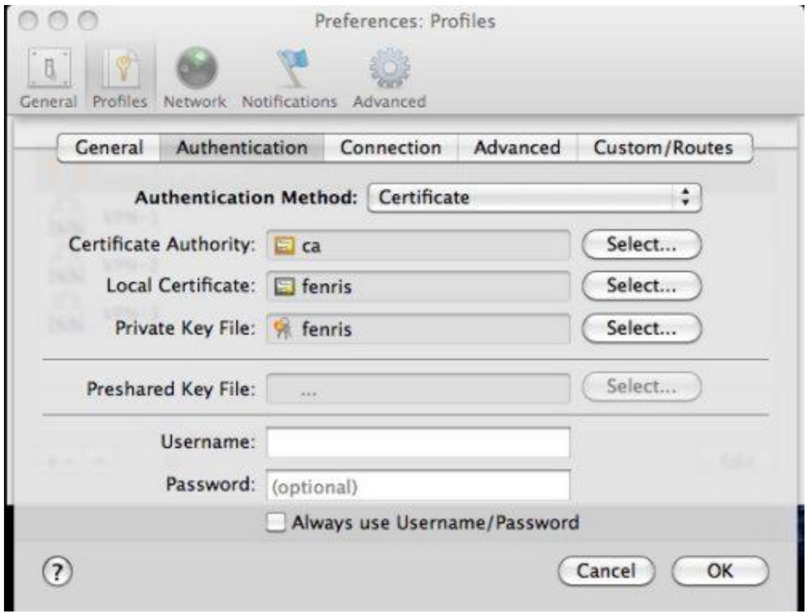
In other tips I've covered how to set up an OpenVPN Linux server and an OpenVPN Linux client. Here, I look at setting up OpenVPN as a client on OS X.

There are a few possible clients to choose from. One popular OpenVPN client for OS X is Tunnelblick. Tunnelblick is free and open source. Another client is Viscosity. It has a cost of $9USD with a 30 day trial. Finally, my client of choice is Shimo, which is not just an OpenVPN client (like the other two), but also works with a number of other VPN and VPN-like solutions: Cisco VPN, IPSec, PPTP/L2TP, SSH, and so forth. Shimo is more expensive than the others, but not by much: it is only €14.95 (about $21USD).

Shimo is also easy to use with OpenVPN. If you have followed along with the other OpenVPN tutorials in this series, you will have a copy of the client certificate, key, and the CA certificate on your system. If not, you will need to obtain them from the server, where they would have been generated, and securely copy (using SSH or a USB disk) them to your computer. Next, start Shimo and head to the Preferences. In the Profiles pane, add a new OpenVPN profile.

Under the General tab, name your new connection -- something like "OpenVPN Home" would suffice. In the Authentication pane, you will need to select your Certificate Authority file (ca.crt), Local Certificate (client.crt), and Private Key File (client.key). Make sure the Authentication Method is set to *Certificate* (**Figure A**). There is no need to set the username and password unless it is required by the server (for the purposes of this series, we elected to use just certificates without further authentication mechanisms).

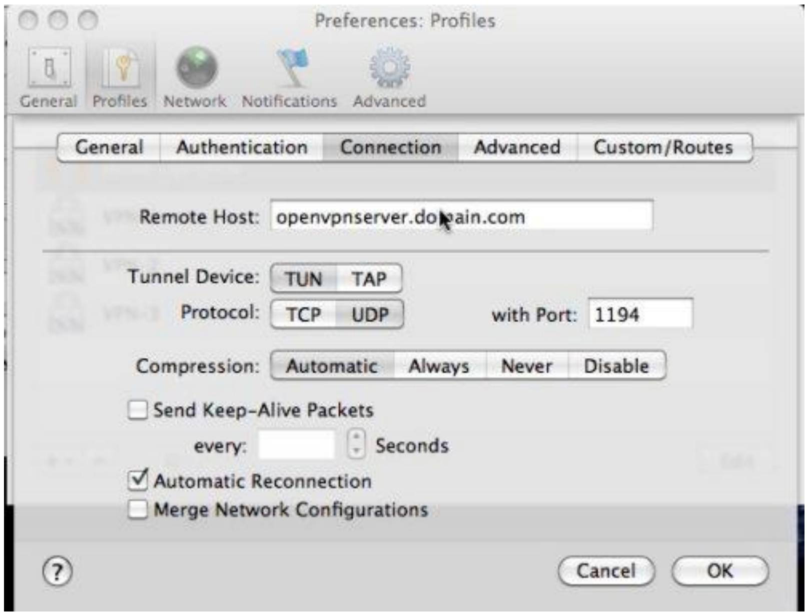
Figure A


Preferences: Profiles

Authentication Method: Certificate
Certificate Authority: ca
Local Certificate: fenris
Private Key File: fenris
Preshared Key File: ...
Username:
Password: (optional)
Always use Username/Password

In the Connection tab, enter in the name of the remote host (i.e., openvpn-server.domain.com). Ensure the Tunnel Device is TUN and the Protocol is UDP (**Figure B**); unless you have changed the connection port on the server, leave it at the default 1194. Set Compression to Automatic, and enable Automatic Reconnection. You can also elect to send keep-alive packets every few seconds to ensure the connection stays up (i.e., maybe send a keep-alive packet ever 120 seconds or so).

Figure B



Remote Host: openvpnserver.domain.com

Tunnel Device: TUN TAP

Protocol: TCP UDP with Port: 1194

Compression: Automatic Always Never Disable

Send Keep-Alive Packets

every: Seconds

☑ Automatic Reconnection

Merge Network Configurations

That's it! You can save the preferences for this profile; go to the Shimo menu icon, and select the new OpenVPN network from the list, and Shimo will establish the connection. If you have enabled the OpenVPN server to push DNS and DNS domain information to clients, when you connect, you will be able to access systems on the remote network by their computer names directly rather than IP addresses.

If you have an iPhone, you're in for an even bigger treat. With iPhone tethering, you can be on the road, anywhere, and securely access the home or work network simply by connecting your iPhone to the laptop (via USB or Bluetooth) and enabling tethering on the iPhone (via Settings | General | Network | Internet Tethering). Once the connection between the Mac and iPhone is established, simply fire up Shimo or whatever OpenVPN client you have chosen, and establish the VPN connection. This works so well that I have been able to obtain a kerberos-ticket and access a kerberos-authentication-only web site on the internal network while sitting in my car across town.

If you only need to use OpenVPN, Shimo may be overkill. It is a fantastic and robust OpenVPN client, but you may wish to give something like Tunnelblick a go first to see if it meets your needs. The latest version of Tunnelblick is 3.0, but it requires you to edit the OpenVPN client configuration directly.

This makes it a lightweight frontend to the OpenVPN command-line program, and the configuration for such can be found in the previous tip about configuring the Linux client. Primarily, you will need to change the "remote" directive to point to the OpenVPN server, and ensure that the ca, cert, and key directives are correct. These directives look for those files in the directory that the configuration file resides in, so you will want to copy those files to ~/Library/Application Support/Tunnelblick/Configurations/.

Once that is done and the configuration file has been saved, use the Tunnelblick menu icon to initiate a connection to the specified OpenVPN server and watch the OpenVPN log output as it connects.

There are a few options to establishing connections to OpenVPN on the Mac. Tunnelblick is good, if a little rough. It is, after all, a simple frontend to the *openvpn* command line program. Shimo is great if you need a little more power, flexibility, and hand-holding. It is also the best of the bunch if you need to connect to different types of VPNs.