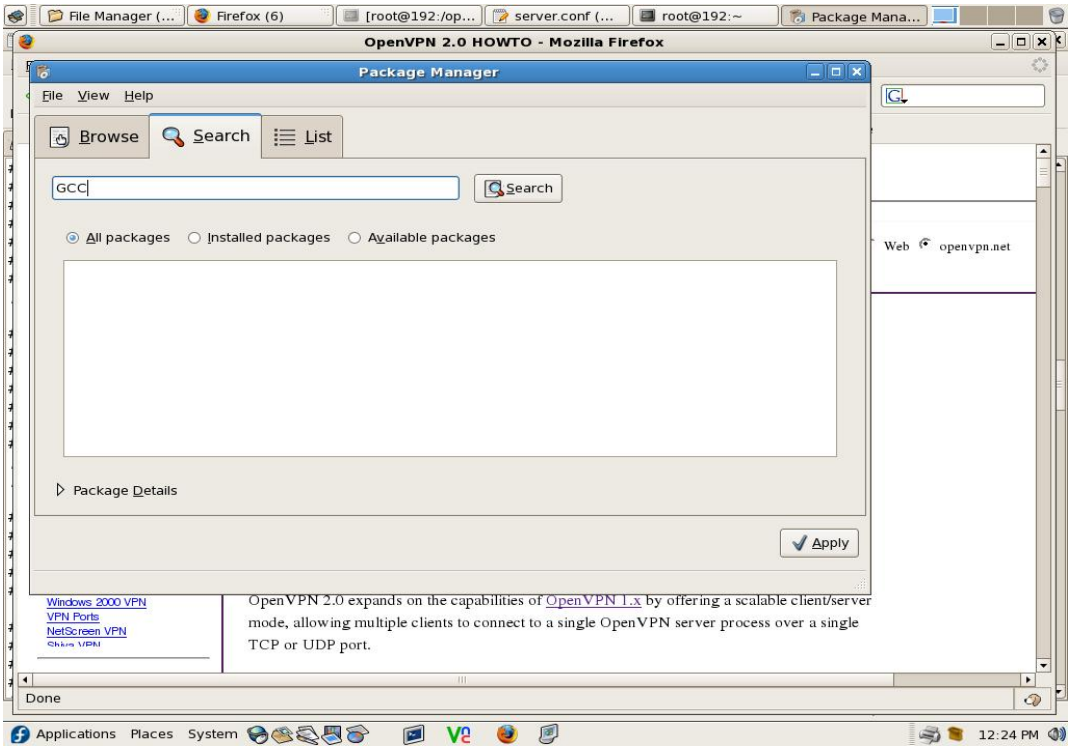
Installing OpenVPN using Fedora Core 6

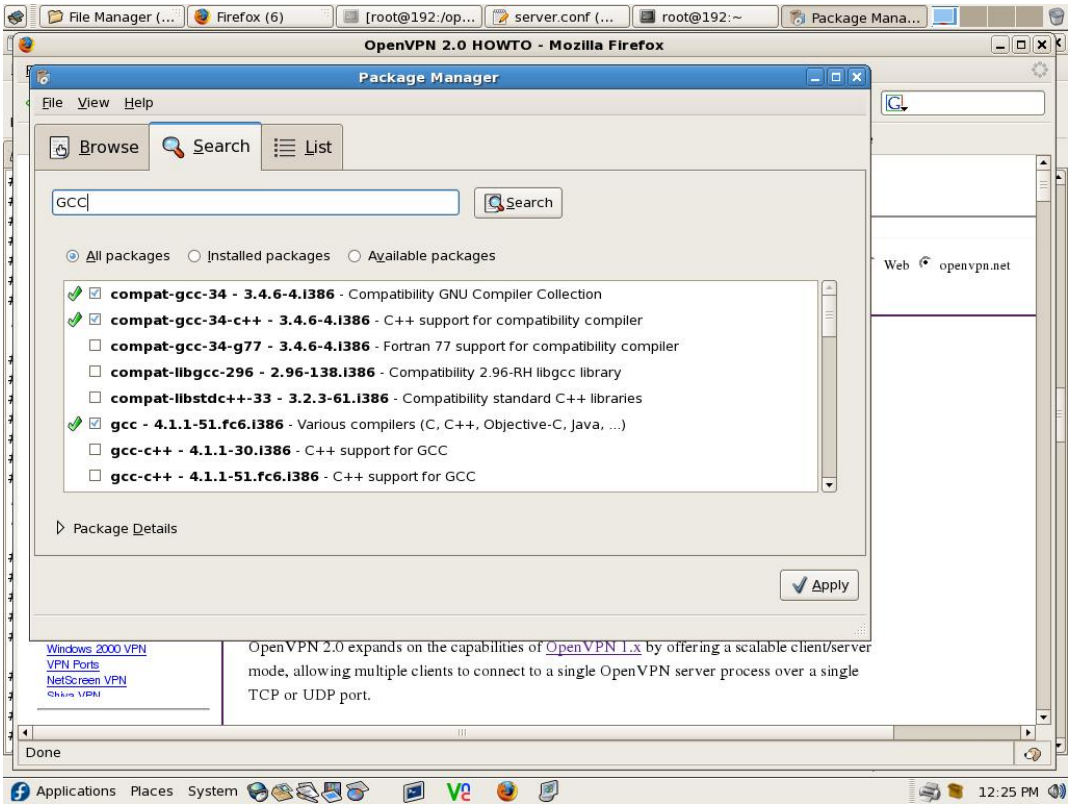
If you are using Debian, Gentoo, or a non-RPM-based Linux distribution, or if you haven't installed or do not wish to install rpm-build, then this installation should work for you. Note, This installation was set up using Fedora Core 6 Linux.

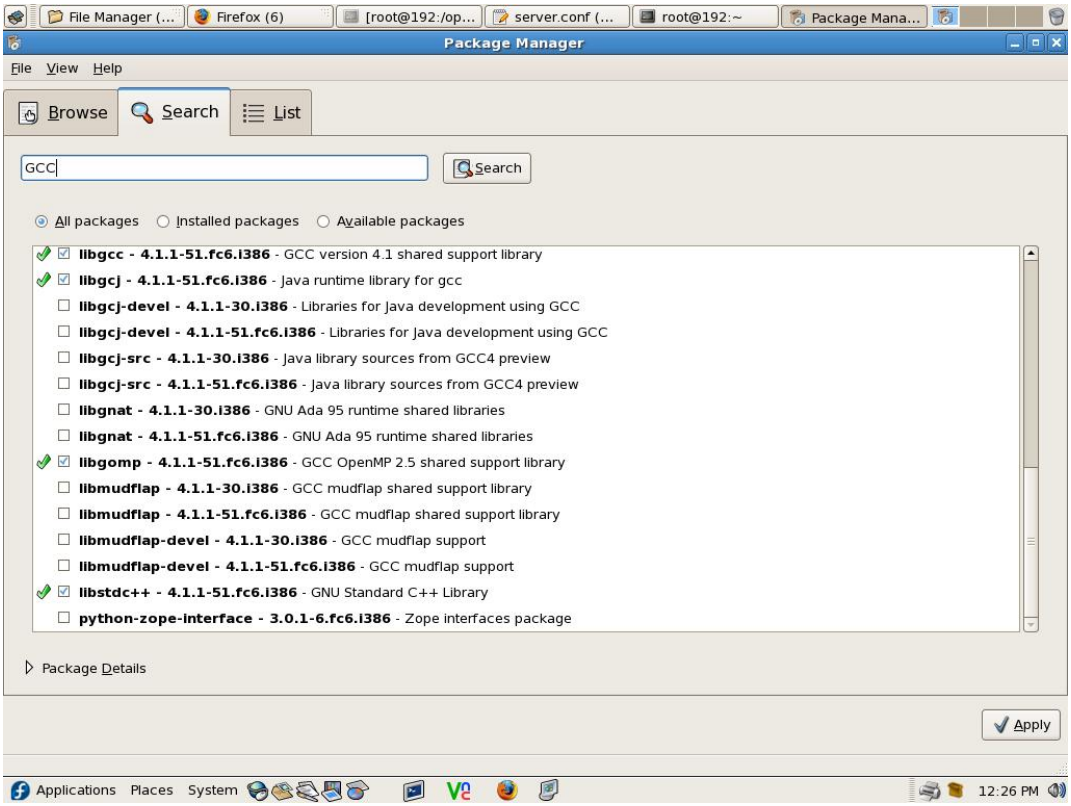
Step 1- Make sure that you have the following GCC packages installed or you will get an error when you attempt to ./configure or make install.

open the Add/Remove software GUI.

Perform a Search for "GCC"




File Manager (...
Firefox (6)
[root@192:/op...
server.conf (...
root@192:~
Package Mana...
OpenVPN 2.0 HOWTO - Mozilla Firefox
Package Manager
File View Help
Browse
Search
List
GCC
Search
All packages
Installed packages
Available packages
compat-gcc-34 - 3.4.6-4.i386 - Compatibility GNU Compiler Collection
compat-gcc-34-c++ - 3.4.6-4.i386 - C++ support for compatibility compiler
compat-gcc-34-g77 - 3.4.6-4.i386 - Fortran 77 support for compatibility compiler
compat-libgcc-296 - 2.96-138.i386 - Compatibility 2.96-RH libgcc library
compat-libstdc++-33 - 3.2.3-61.i386 - Compatibility standard C++ libraries
gcc - 4.1.1-51.fc6.i386 - Various compilers (C, C++, Objective-C, Java, ...)
gcc-c++ - 4.1.1-30.i386 - C++ support for GCC
gcc-c++ - 4.1.1-51.fc6.i386 - C++ support for GCC
Package Details
Apply
Web
openvpn.net
Windows 2000 VPN
VPN Ports
NetScreen VPN
OpenVPN 2.0 expands on the capabilities of OpenVPN 1.x by offering a scalable client/server mode, allowing multiple clients to connect to a single OpenVPN server process over a single TCP or UDP port.
Done
Applications
Places
System
12:25 PM

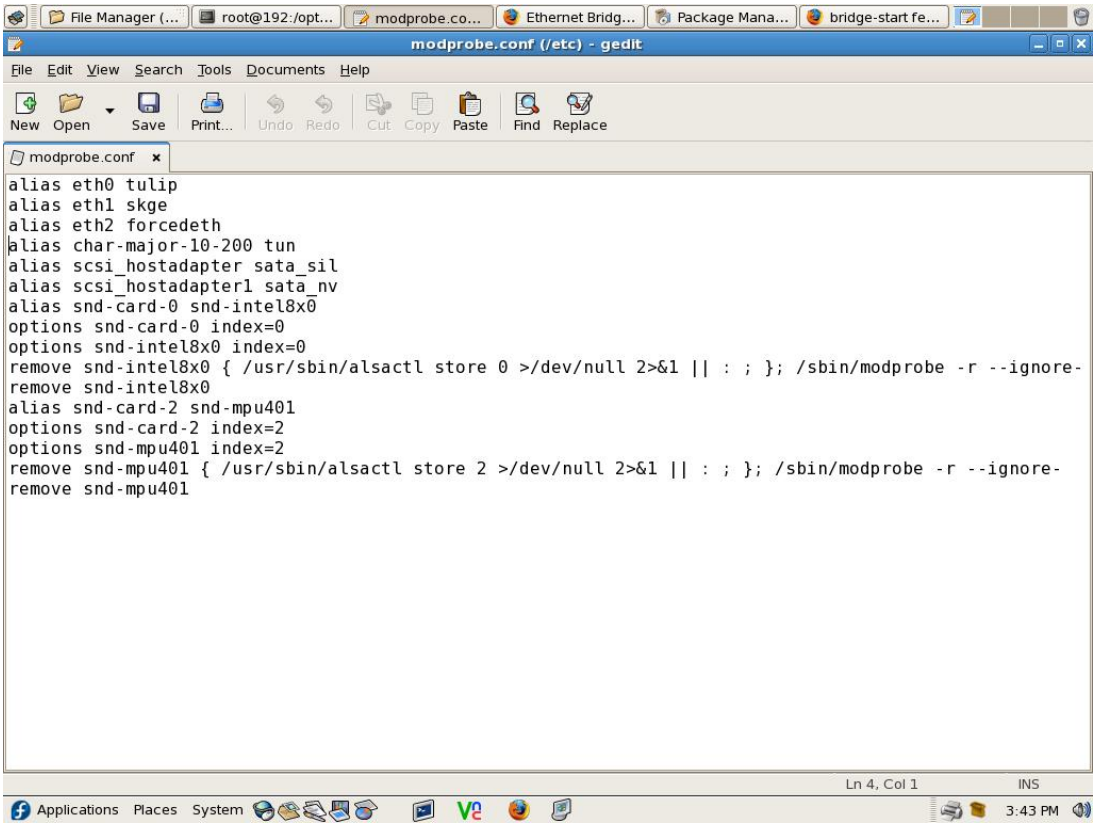

File Manager (...
Firefox (6)
[root@192:/op...
server.conf (...
root@192:~
Package Mana...
Package Manager
File View Help
Browse
Search
List
GCC
Search
All packages
Installed packages
Available packages
libgcc - 4.1.1-51.fc6.i386 - GCC version 4.1 shared support library
libgcj - 4.1.1-51.fc6.i386 - Java runtime library for gcc
libgcj-devel - 4.1.1-30.i386 - Libraries for Java development using GCC
libgcj-devel - 4.1.1-51.fc6.i386 - Libraries for Java development using GCC
libgcj-src - 4.1.1-30.i386 - Java library sources from GCC4 preview
libgcj-src - 4.1.1-51.fc6.i386 - Java library sources from GCC4 preview
libgnat - 4.1.1-30.i386 - GNU Ada 95 runtime shared libraries
libgnat - 4.1.1-51.fc6.i386 - GNU Ada 95 runtime shared libraries
libgomp - 4.1.1-51.fc6.i386 - GCC OpenMP 2.5 shared support library
libmudflap - 4.1.1-30.i386 - GCC mudflap shared support library
libmudflap - 4.1.1-51.fc6.i386 - GCC mudflap shared support library
libmudflap-devel - 4.1.1-30.i386 - GCC mudflap support
libmudflap-devel - 4.1.1-51.fc6.i386 - GCC mudflap support
libstdc++ - 4.1.1-51.fc6.i386 - GNU Standard C++ Library
python-zope-interface - 3.0.1-6.fc6.i386 - Zope interfaces package
Package Details
Apply
Applications
Places
System
12:26 PM

Step 2- Download the tarball file from Openvpn.net

(the file is currently named: **openvpn-2.0.9.tar.gz)**

Next: Find the file modprobe.conf usually located in /etc. You can locate this file using the Terminal Root> locate modprobe.conf

Now edit this file by adding to it the line: "alias char-major-10-200 tun" and save.
This will automatically start your tunneling interface every time Linux starts.



Step 3- Using Terminal, enter this command to extract the tarball contents into the default openvpn folder. (It will create the folder for you)

Root> tar xfz openvpn-2.0.9.tar.gz or tar -zxvf openvpn-2.0.9.tar.gz

Then cd to the newly created directory and type:

```
Root>   ./configure --disable-lzo
Root>   make
Root>   make install
```

I am using –disable-lzo because I received an error and it told me that I may need to disable lzo. FYI, disabling lzo disables compression, which is ok to do.

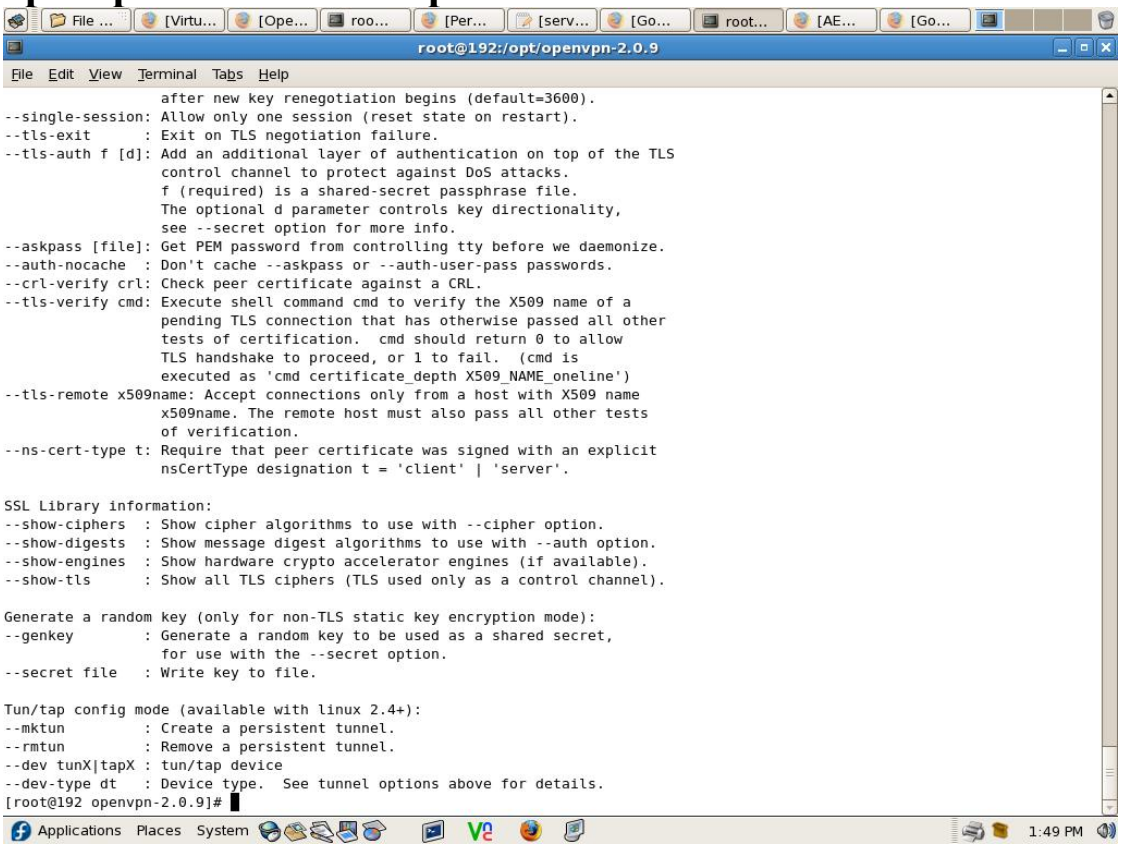
Step 4- using Terminal, move the newly created folder openvpn-2.0.9 to the opt directory

Root> mv openvpn-2.09 /opt

This folder will now be located in /opt/openvpn-2.0.9

Step 5- Using Terminal make sure you are in the /opt/openvpn-2.0.9 directory. Enter this to test the Openvpn installation. Root> openvpn

-If this command works, you will see numerous options listed for use with openvpn. See the example window below.



Please Read the following before we get started with the configuration.

Numbering private subnets

Setting up a VPN often entails linking together private subnets from different locations.

The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets (codified in RFC 1918):

10.0.0.0	10.255.255.255	(10/8 prefix)
172.16.0.0	172.31.255.255	(172.16/12 prefix)
192.168.0.0	192.168.255.255	(192.168/16 prefix)

While addresses from these netblocks should normally be used in VPN configurations, it's important to select addresses that minimize the probability of IP address or subnet conflicts. The types of conflicts that need to be avoided are:

- conflicts from different sites on the VPN using the same LAN subnet numbering, or
- remote access connections from sites which are using private subnets which conflict with your VPN subnets.

For example, suppose you use the popular 192.168.0.0/24 subnet as your private LAN subnet. Now you are trying to connect to the VPN from an internet cafe which is using the same subnet for its WiFi LAN. You will have a routing conflict because your machine won't know if 192.168.0.1 refers to the local WiFi gateway or to the same address on the VPN.

As another example, suppose you want to link together multiple sites by VPN, but each site is using 192.168.0.0/24 as its LAN subnet. This won't work without adding a complexifying layer of NAT translation, because the VPN won't know how to route packets between multiple sites if those sites don't use a subnet which uniquely identifies them.

The best solution is to avoid using 10.0.0.0/24 or 192.168.0.0/24 as private LAN network addresses. Instead, use something that has a lower probability of being used in a WiFi cafe, airport, or hotel where you might expect to connect from remotely. The best candidates are subnets in the middle of the vast 10.0.0.0/8 netblock (for example 10.66.77.0/24).

And to avoid cross-site IP numbering conflicts, always use unique numbering for your LAN subnets.

Step 6- Setting up your own Certificate Authority (CA) and generating certificates and keys for an OpenVPN server and multiple clients

Overview

The first step in building an OpenVPN 2.0 configuration is to establish a PKI (public key infrastructure). The PKI consists of:

- a separate certificate (also known as a public key) and private key for the server and each client, and
- a master Certificate Authority (CA) certificate and key which is used to sign each of the server and client certificates.

OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established.

Both server and client will authenticate the other by first verifying that the presented certificate was signed by the master certificate authority (CA), and then by testing information in the now-authenticated certificate header, such as the certificate common name or certificate type (client or server).

This security model has a number of desirable features from the VPN perspective:

- The server only needs its own certificate/key -- it doesn't need to know the individual certificates of every client which might possibly connect to it.
- The server will only accept clients whose certificates were signed by the master CA certificate (which we will generate below). And because the server can perform this signature verification without needing access to the CA private key itself, it is possible for the CA key (the most sensitive key in the entire PKI) to reside on a completely different machine, even one without a network connection.
- If a private key is compromised, it can be disabled by adding its certificate to a CRL (certificate revocation list). The CRL allows compromised certificates to be selectively rejected without requiring that the entire PKI be rebuilt.
- The server can enforce client-specific access rights based on embedded certificate fields, such as the Common Name.

Next: Generate the master Certificate Authority (CA) certificate & key

In this section we will generate a master CA certificate/key, a server certificate/key, and certificates/keys for 3 separate clients.

For PKI management, we will use a set of scripts bundled with OpenVPN.

cd to the **easy-rsa** subdirectory of the OpenVPN distribution. /opt/openvpn-2.0.9/easy-rsa

Now edit the **vars** file and set the KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG, and KEY_EMAIL parameters. Don't leave any of these parameters blank.

Next, initialize the PKI.:

```
. ./vars
./clean-all
./build-ca
```

The final command (**build-ca**) will build the certificate authority (CA) certificate and key by invoking the interactive **openssl** command:

```
-----------------------------------------------------------------OUTPUT--------
ai:easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
............++++++
............++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [KG]:
State or Province Name (full name) [NA]:
Locality Name (eg, city) [BISHKEK]:
Organization Name (eg, company) [OpenVPN-TEST]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:OpenVPN-CA
Email Address [me@myhost.mydomain]:
```

Note that in the above sequence, most queried parameters were defaulted to the values set in the **vars** or **vars.bat** files. The only parameter which must be explicitly entered is the **Common Name**. In the example above, I used "OpenVPN-CA".

Step 7- Next: Generate certificate & key for server

Next, we will generate a certificate and private key for the server. On Linux/BSD/Unix:

```
./build-key-server server
```

As in the previous step, most parameters can be defaulted. When the **Common Name** is queried, enter "server". Two other queries require positive responses, "Sign the certificate? [y/n]" and "1 out of 1 certificate requests certified, commit? [y/n]".

Generate certificates & keys for 3 clients

Generating client certificates is very similar to the previous step. On Linux/BSD/Unix:

```
./build-key client1
./build-key client2
./build-key client3
```

If you would like to password-protect your client keys, substitute the **build-key-pass** script.

Remember that for each client, make sure to type the appropriate **Common Name** when prompted, i.e. "client1", "client2", or "client3". Always use a unique common name for each client.

Generate Diffie Hellman parameters

Diffie Hellman parameters must be generated for the OpenVPN server.

```
./build-dh
```

```
--------------------------------------------------OUTPUT-------------
ai:easy-rsa # ./build-dh
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.................+.......................................
...................+............+.................+........
```

.....

Key Files

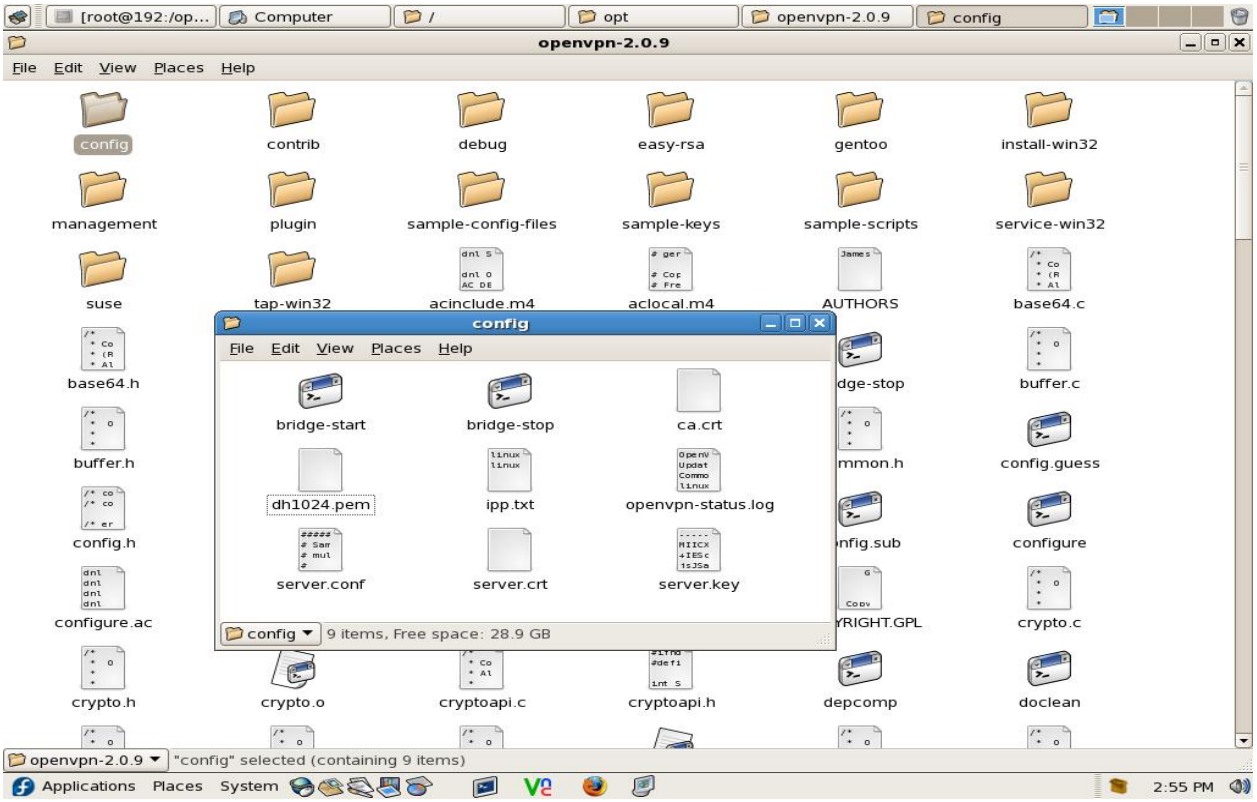
Now we will find our newly-generated keys and certificates in the **keys** subdirectory. Here is an explanation of the relevant files:

Filename	Needed By	Purpose	Secret
ca.crt	server + all clients	Root CA certificate	NO
ca.key	key signing machine only	Root CA key	YES
dh{n}.pem	server only	Diffie Hellman parameters	NO
server.crt	server only	Server Certificate	NO
server.key	server only	Server Key	YES
client1.crt	client1 only	Client1 Certificate	NO
client1.key	client1 only	Client1 Key	YES

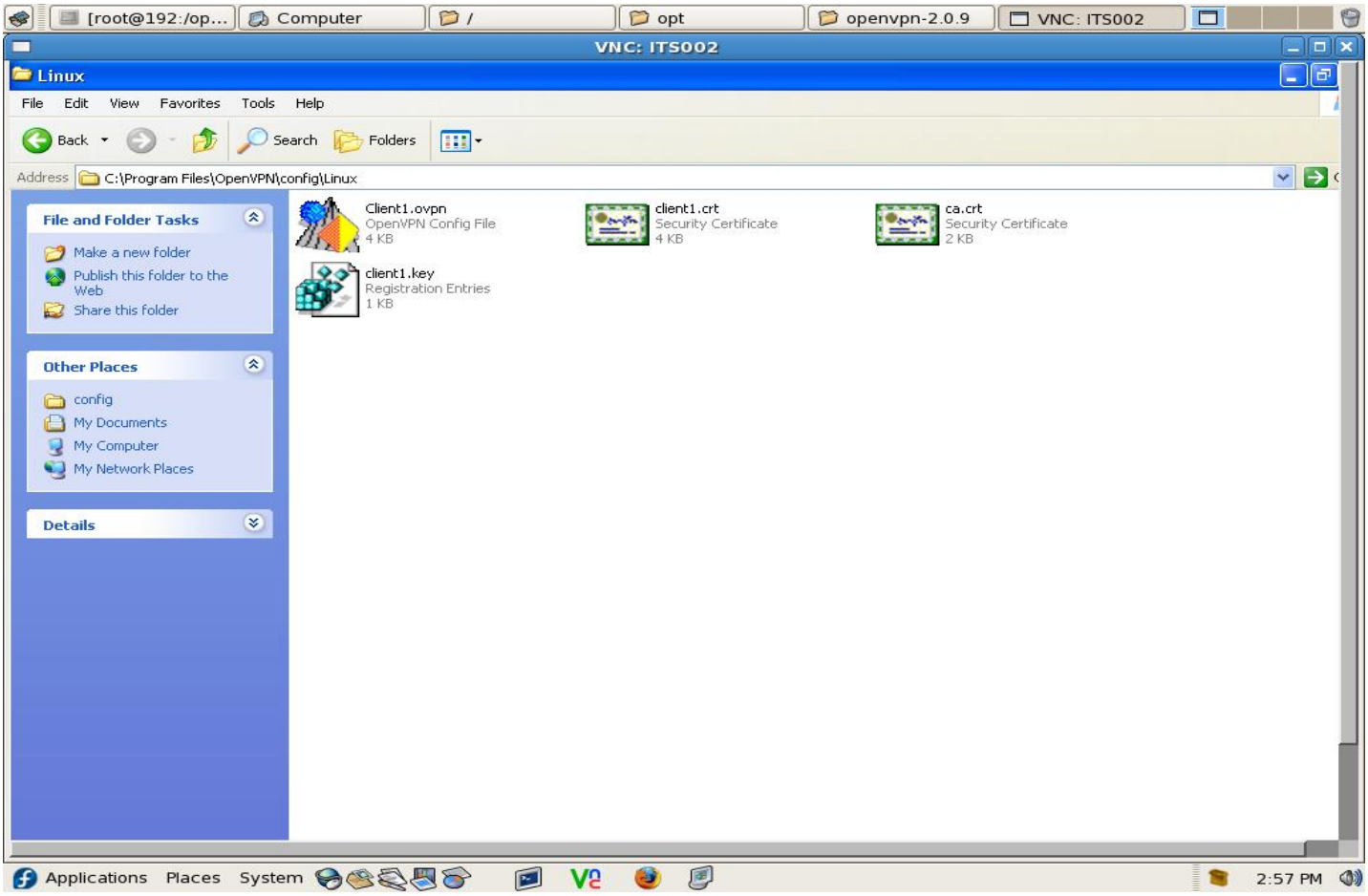
client2.crt	client2 only	Client2 Certificate	NO
client2.key	client2 only	Client2 Key	YES
client3.crt	client3 only	Client3 Certificate	NO
client3.key	client3 only	Client3 Key	YES

The final step in the key generation process is to copy all files to the machines which need them, taking care to copy secret files over a secure channel. They are to be placed in the OpenVPN/sample-config directory on client and server. Note, I created a config directory located within OpenVpn-2.0.9/config.

This is my server config folder and the necessary files. Note that the bridge-start and bridge-stop script files are located in the sample-scripts folder but are not needed unless you decide to utilize the network bridge option.



This is my client config folder and the necessary files.



STEP 8 - Create configuration files for the server

Getting the sample config files

It's best to use the OpenVPN sample configuration files as a starting point for your own configuration. These files can also be found in

- the **sample-config-files** directory of the OpenVPN source distribution will be located in the /opt/openvpn.2.0.9 directory. Further note, if you like, you can just keep these files in the sample-config-files directory, but in this example I created a config directory. Openvpn-2.0.9/config

Note that on Linux, the sample configuration files are named **server.conf** and **client.conf**. On Windows they are named **server.ovpn** and **client.ovpn**.

Editing the server configuration file "server.conf"

This example is specifically for creating a routed IP tunnel using OPENVPN DHCP…WITHOUT utilizing bridging. This basically means that your VPN traffic will communicate on a completely separate private network, resulting in your OPENVPN clients being unable to communicate with a regular client on your home/office network.

```
# Sample OpenVPN 2.0 config file for
# multi-client server.
# #
# This file is for the server side
# of a many-clients <-> one-server
# OpenVPN configuration.
# #
# OpenVPN also supports
# single-machine <-> single-machine
# configurations (See the Examples page
# on the web site for more info).
# #
# This config should work on Windows
# or Linux/BSD systems. Remember on
# Windows to quote pathnames and use
# double backslashes, e.g.:
# "C:\\Program Files\\OpenVPN\\config\\foo.key"
```

```
#Comments are proceeded with # or ;

# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one. You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
;proto tcp
proto udp

# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an Ethernet tunnel.
# Use "dev tap0" if you are Ethernet bridging
```

```
# and have precreated a tap0 virtual interface
# and bridged it with your ethernet interface.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
# Use "dev tap0" if you are Ethernet bridging
;dev tap0
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one. On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key). Each client
# and the server must have their own cert and
# key file. The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys. Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ca.crt
cert server.crt
key server.key # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
# openssl dhparam -out dh1024.pem 1024
# Substitute 2048 for 1024 if you are using
```

```
# 2048 bit keys.
dh dh1024.pem

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1.
#Comment this line out if you are Ethernet bridging.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file. If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
# NIC interface. Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 192.168.1.101/255.255.255.0. Finally we
# must set aside an IP range in this subnet
# (start=192.168.1.240 end=192.168.1.250) to allocate
# to connecting clients.
# Leave this line commented out unless you are Ethernet bridging.
;server-bridge 192.168.1.101 255.255.255.0 192.168.1.240 192.168.1.250

# Push routes to the client to allow it
# to reach other private subnets behind
# the server. Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.1.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).
```

```
# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
# iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN. This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
# ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
# of clients. There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
# group, and firewall the TUN/TAP interface
# for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
# modify the firewall in response to access
# from different clients. See man
# page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# the TUN/TAP interface to the internet in
# order for this to work properly).
# CAVEAT: May break client's network config if
# client's local DHCP server packets get routed
# through the tunnel. Solution: make sure
```

```
# client's local DHCP server is reachable via
# a more specific route than the default route
# of 0.0.0.0/0.0.0.0.
;push "redirect-gateway"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses. CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
;push "dhcp-option DNS 10.8.0.1"
;push "dhcp-option WINS 10.8.0.1"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
client-to-client

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names. This is recommended
# only for testing purposes. For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
```

```
# Generate with:
# openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC # Blowfish (default)
;cipher AES-128-CBC # AES
;cipher DES-EDE3-CBC # Triple-DES

# Enable compression on the VPN link.
#We are disabling this because we couldn't install this feature! If you disable it here, you must also
# disable it in the client config file.
;comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nobody

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
```

```
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it. Use one
# or the other (but not both).
;log openvpn.log
;log-append openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3
# Silence repeating messages. At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20
```

-END of SERVER.CONF file

The sample server configuration file is an ideal starting point for an OpenVPN server configuration. It will create a VPN using a virtual **TUN** network interface (for routing), will listen for client connections on **UDP port 1194**, and will distribute virtual addresses to connecting clients from the **10.8.0.0/24** subnet. **Note that Port 1194 is OpenVpn's official port number although you may decide to use a different port.**

*** Before you use the sample configuration file, you should first edit the **ca**, **cert**, **key**, and **dh** parameters to point to the files you generated in the PKI section above.

At this point, the server configuration file is usable, however you still might want to customize it further:

- If you are using Ethernet bridging, you must use **server-bridge** and **dev tap0** instead of **server** and **dev tun**.
- If you want your OpenVPN server to listen on a TCP port instead of a UDP port, use **proto tcp** instead of **proto udp** (If you want OpenVPN to listen on both a UDP and TCP port, you must run two separate OpenVPN instances).
- If you want to use a virtual IP address range other than **10.8.0.0/24**, you should modify the **server** directive. Remember that this virtual IP address range should be a private range which is currently unused on your network.
- Uncomment out the **client-to-client** directive if you would like connecting clients to be able to reach each other over the VPN. By default, clients will only be able to reach the server.
- If you are using Linux, BSD, or a Unix-like OS, you can improve security by uncommenting out the

user nobody and **group nobody** directives.

If you want to run multiple OpenVPN instances on the same machine, each using a different configuration file, it is possible if you:

- Use a different **port** number for each instance (the UDP and TCP protocols use different port spaces so you can run one daemon listening on UDP-1194 and another on TCP-1194).
- If you are using Windows, each OpenVPN configuration needs to have its own TAP-Win32 adapter. You can add additional adapters by going to **Start Menu -> All Programs -> OpenVPN -> Add a new TAP-Win32 virtual ethernet adapter**.
- If you are running multiple OpenVPN instances out of the same directory, make sure to edit directives which create output files so that multiple instances do not overwrite each other's output files. These directives include **log**, **log-append**, **status**, and **ifconfig-pool-persist**.

STEP 9 - Editing the client configuration files. This file is named client1.conf or client1.ovpn This is an example of my working client-side config file. In this example I am using a Windows XP client.

```
# Sample client-side OpenVPN 2.0 config file
# for connecting to multi-client server.
#
# This configuration can be used by multiple
# clients, however each client should have
# its own cert and key files.
#
# On Windows, you might want to rename this
# file so it has a .ovpn extension
```

```
# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
#Use Dev TAP only if you are using Ethernet-bridging from the OpenVpn server
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
```

```
# from the Network Connections panel
# if you have more than one.  On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
#I renamed my virtual Ethernet TAP adapter to MYTAP
dev-node MYTAP

# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
;proto tcp
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote Remote Server IP Address 1194
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
;user nobody
;group nobody

# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
```

```
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca ca.crt
cert Client1.crt
key Client1.key

# Verify server certificate by checking
# that the certicate has the nsCertType
# field set to "server".  This is an
# important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server".  The build-key-server
# script in the easy-rsa folder will do this.
ns-cert-type server

# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
;cipher x

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
;comp-lzo
```

```
# Set log file verbosity.
verb 3

# Silence repeating messages
;mute 20
```

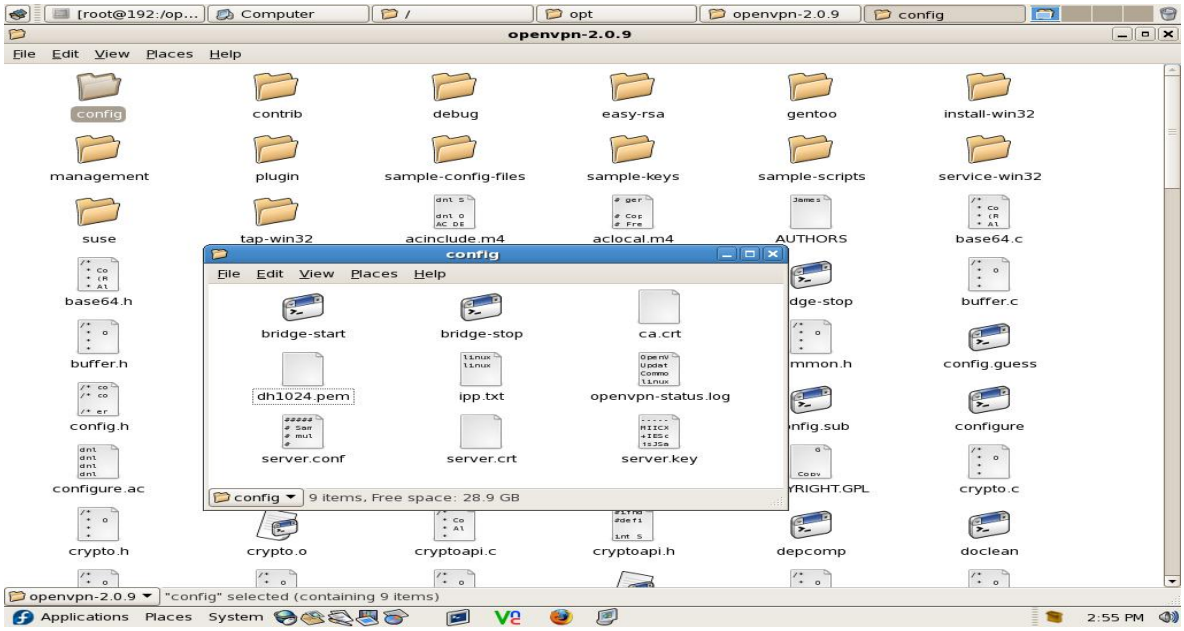
END of CLIENT1.CONF or CLIENT1.OVPN File

The sample client configuration file (**client.conf** on Linux/BSD/Unix or **client.ovpn** on Windows) mirrors the default directives set in the sample server configuration file.

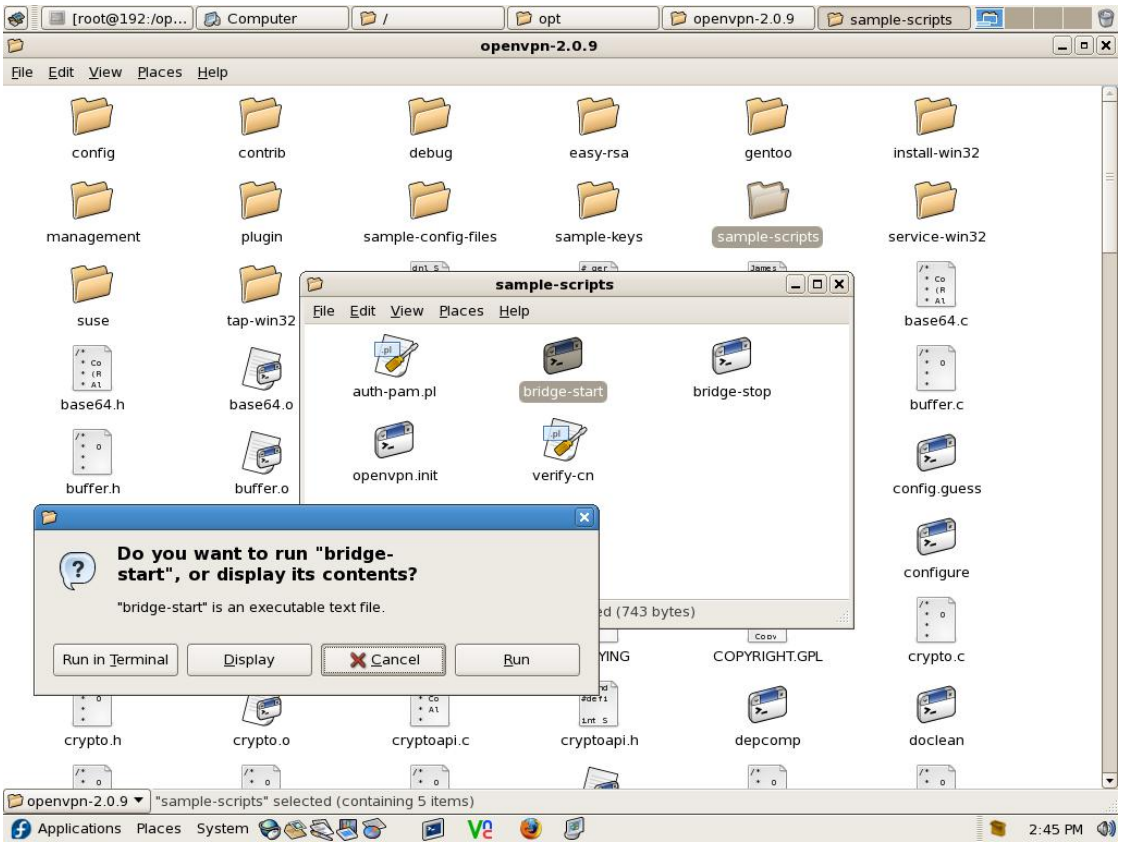
- Like the server configuration file, first edit the **ca**, **cert**, and **key** parameters to point to the files you generated in the PKI section above. Note that each client should have its own **cert**/**key** pair. Only the **ca** file is universal across the OpenVPN server and all clients.
- Next, edit the **remote** directive to point to the hostname/IP address and port number of the OpenVPN server (if your OpenVPN server will be running on a single-NIC machine behind a firewall/NAT-gateway, use the public IP address of the gateway, and a port number which you have configured the gateway to forward to the OpenVPN server).
- Finally, ensure that the client configuration file is consistent with the directives used in the server configuration. The major thing to check for is that the **dev** (tun or tap) and **proto** (udp or tcp) directives are consistent. Also make sure that **comp-lzo** and **fragment**, if used, are present in both client and server config files.

STEP 10A – SKIP THIS STEP IF YOU ARE NOT ETHERNET BRIDGING!!

Ethernet Bridging. First, edit both the server.conf and client.ovpn configuration files as stated above in order to get ehternet-bridging to work. Second, open the sample-scripts folder located in the openvpn-2.0.9/sample-scripts and copy the files “server-start” and “server-stop” to your config folder that you made earlier, located in /opt/openvpn-2.0.9/config. Your config folder should contain these files.

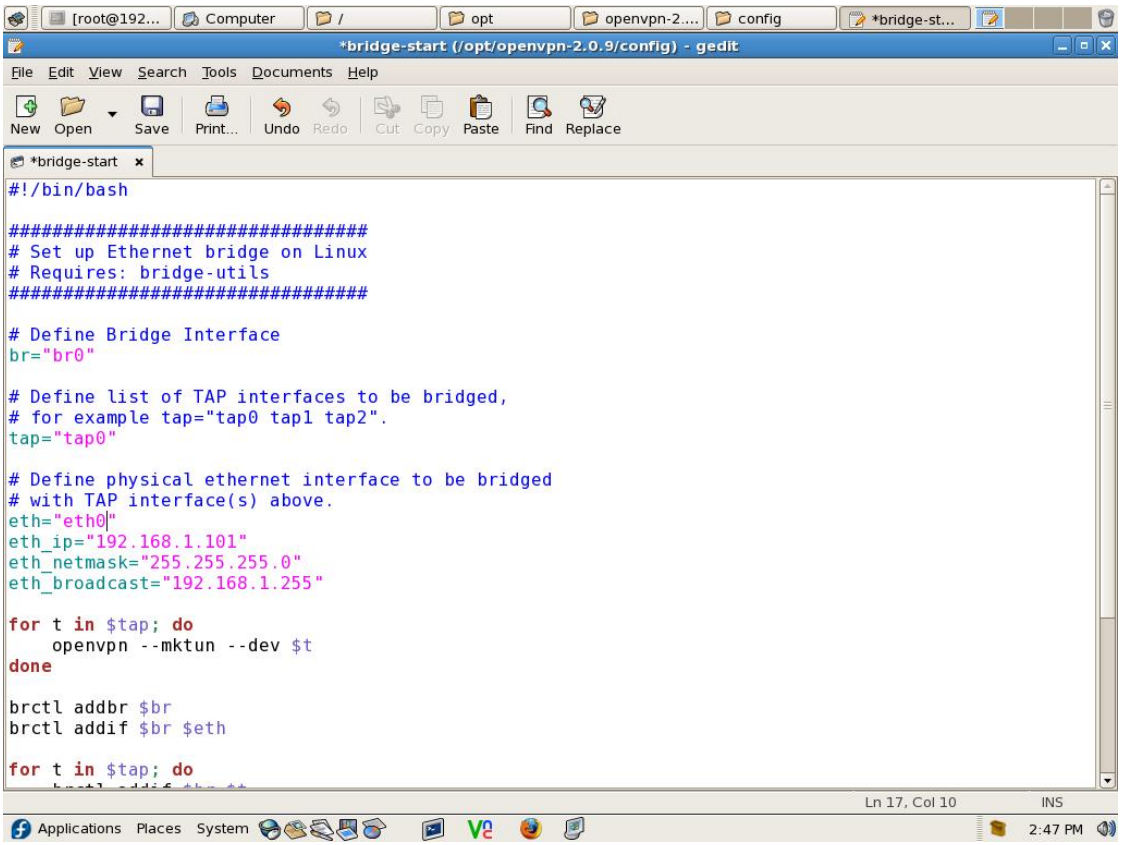


Now you must edit “bridge-start” using your favorite editor like “Gedit”



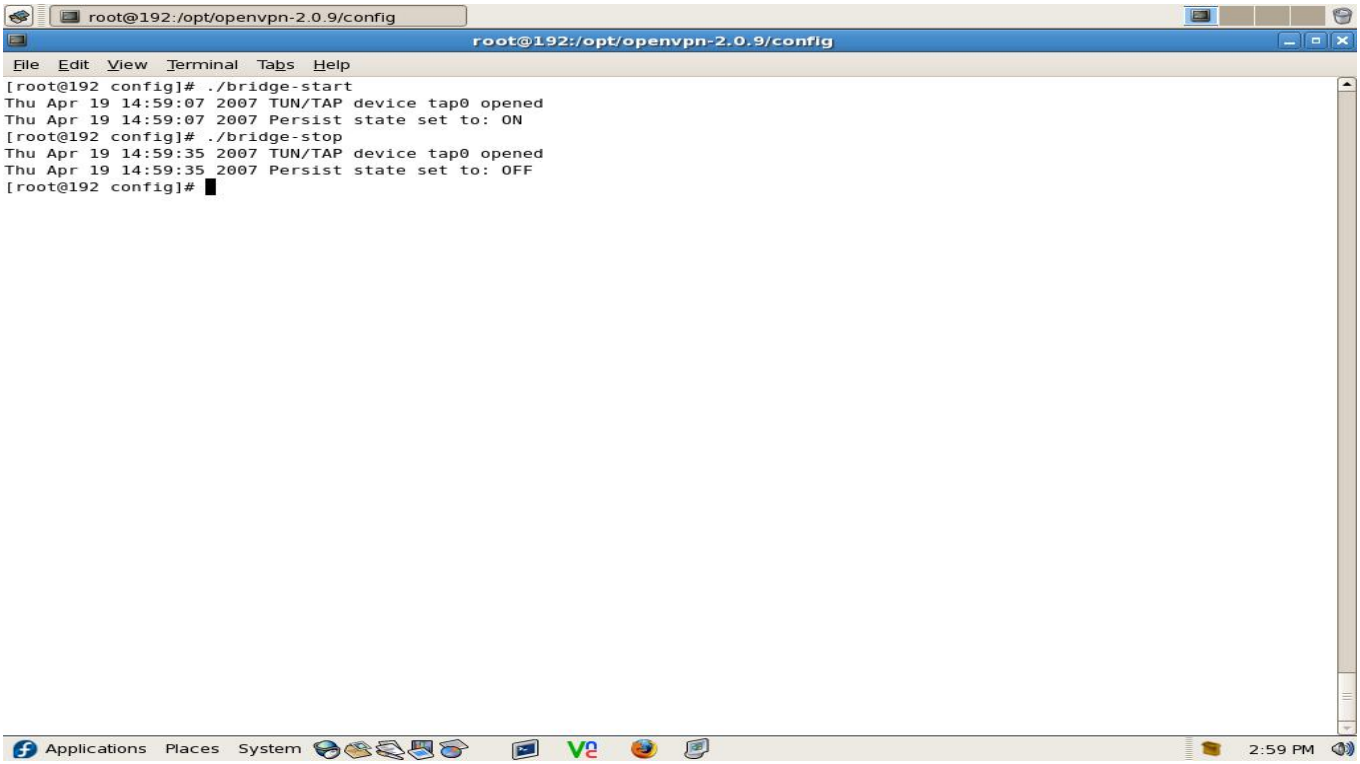
Now edit the bridge-start file accordingly. Note that your active Ethernet adapter may be ETH1 or ETH2 and that your local home or office network address and

subnet mask may be different.



Save your changes and start and stop the bridge support by typing this in the terminal/command line.

Root> ./bridge-start



or Root> ./bridge-stop

to stop the bridge adapter.

STEP 10B -Start up the VPN and test for initial connectivity

Starting the server using the Terminal

First, make sure the OpenVPN server will be accessible from the internet. That means:

- opening up UDP port 1194 on the firewall (or whatever TCP/UDP port you've configured), or
- setting up a port forward rule to forward UDP port 1194 from the firewall/gateway to the machine running the OpenVPN server.

Next, make sure that the TUN/TAP interface is not firewalled.

To simplify troubleshooting, it's best to initially start the OpenVPN server from the command line (or right-click on the **.ovpn** file on Windows), rather than start it as a daemon or service:

```
/opt/openvpn.2.0.9/config> openvpn server.conf
```

A normal server startup should look like this (output will vary across platforms):

----------OUTPUT----------

```
Sun Feb  6 20:46:38 2005 OpenVPN 2.0_rc12 i686-suse-linux [SSL] [LZO] [EPOLL] built on
Feb  5 2005
Sun Feb  6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key
Sun Feb  6 20:46:38 2005 TLS-Auth MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Sun Feb  6 20:46:38 2005 TUN/TAP device tun1 opened
Sun Feb  6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2 mtu 1500
Sun Feb  6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask 255.255.255.0 gw 10.8.0.2
Sun Feb  6 20:46:38 2005 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:23 ET:0 EL:0
AF:3/1 ]
Sun Feb  6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194
Sun Feb  6 20:46:38 2005 UDPv4 link remote: [undef]
Sun Feb  6 20:46:38 2005 MULTI: multi_init called, r=256 v=256
Sun Feb  6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62
Sun Feb  6 20:46:38 2005 IFCONFIG POOL LIST
Sun Feb  6 20:46:38 2005 Initialization Sequence Completed
```

Starting the client

As in the server configuration, it's best to initially start the OpenVPN server from the command line (or on Windows, by right-clicking on the **client.ovpn** file), rather than start it as a daemon or service:

From the Windows Command Prompt, go to the "program files/openvpn/config" folder and type: **openvpn client1.ovpn**

or LINUX go to the /opt/openvpn-2.0.9/config:

openvpn client1.conf

A normal client startup on Windows will look similar to the server output above, and should end with the **Initialization Sequence Completed** message.

Now, try a ping across the VPN from the client. If you are using routing (i.e. **dev tun** in the server config file), try:

ping 10.8.0.1

If you are using bridging (i.e. **dev tap** in the server config file), try to ping the IP address of a machine on the server's ethernet subnet.

Ping 192.168.1.27

(This is just some random client on your office/home private network)

If the ping succeeds, congratulations! You now have a functioning VPN.

Troubleshooting

If the ping failed or the OpenVPN client initialization failed to complete, here is a checklist of common

symptoms and their solutions:

- You get the error message: **TLS Error: TLS key negotiation failed to occur within 60 seconds (check your network connectivity)**. This error indicates that the client was unable to establish a network connection with the server.

  Solutions:

  - Make sure the client is using the correct hostname/IP address and port number which will allow it to reach the OpenVPN server.
  - If the OpenVPN server machine is a single-NIC box inside a protected LAN, make sure you are using a correct port forward rule on the server's gateway firewall. For example, suppose your OpenVPN box is at 192.168.4.4 inside the firewall, listening for client connections on UDP port 1194. The NAT gateway servicing the 192.168.4.x subnet should have a port forward rule that says **forward UDP port 1194 from my public IP address to 192.168.4.4**.
  - Open up the server's firewall to allow incoming connections to UDP port 1194 (or whatever TCP/UDP port you have configured in the server config file).
- You get the error message: **Initialization Sequence Completed with errors** -- This error can occur on Windows if (a) You don't have the DHCP client service running, or (b) You are using certain third-party personal firewalls on XP SP2.

  Solution: Start the DHCP client server and make sure that you are using a personal firewall which is known to work correctly on XP SP2.

- You get the **Initialization Sequence Completed** message but the ping test fails -- This usually indicates that a firewall on either server or client is blocking VPN network traffic by filtering on the TUN/TAP interface.

  Solution: Disable the client firewall (if one exists) from filtering the TUN/TAP interface on the client. For example on Windows XP SP2, you can do this by going to **Windows Security Center -> Windows Firewall -> Advanced** and unchecking the box which corresponds to the TAP-Win32 adapter (disabling the client firewall from filtering the TUN/TAP adapter is generally reasonable from a security perspective, as you are essentially telling the firewall not to block authenticated VPN traffic). Also make sure that the TUN/TAP interface on the server is not being filtered by a firewall (having said that, note that selective firewalling of the TUN/TAP interface on the server side can confer certain security benefits. See the access policies section below).

- The connection stalls on startup when using a **proto udp** configuration, the server log file shows this line:

  ```
  TLS: Initial packet from x.x.x.x:x, sid=xxxxxxxx xxxxxxxx
  ```

  however the client log does not show an equivalent line.

  Solution: You have a one-way connection from client to server. The server to client direction is blocked by a firewall, usually on the client side. The firewall can either be (a) a personal software

firewall running on the client, or (b) the NAT router gateway for the client. Modify the firewall to allow returning UDP packets from the server to reach the client.

See the FAQ for additional troubleshooting information.

Configuring OpenVPN to run automatically on system startup

The lack of standards in this area means that most OSes have a different way of configuring daemons/services for autostart on boot. The best way to have this functionality configured by default is to install OpenVPN as a package, such as via RPM on Linux or using the Windows installer.

Linux

If you install OpenVPN via an RPM package on Linux, the installer will set up an **initscript**. When executed, the initscript will scan for **.conf** configuration files in **/etc/openvpn**, and if found, will start up a separate OpenVPN daemon for each file.