Secure Boot CDs for VPN HOWTO

Table of Contents

Secure Boot CDs for VPN HOWTO

Jeffery Douglas Waddell jefferydouglaswaddell (at) gmail (dot) com

V1.0 2007−05−15 Initial release

This document describes the creation of live boot CDs used to make secure VPN connections from anywhere (over the Internet) to internal networks that have firewall exposure to the Internet.

---

1. Introduction

For several years now, users at the Institution where I currently work part−time have expressed a need for a robust and secure connection to the internal network. The internal network actually has some clients that are on private networks and some clients that are Internet−routeable.

Being mostly a Microsoft shop, the options that have been available were not deemed appropriate for various reasons, usually due to security concerns. After being introduced to the problem a few months ago and listening to the concerns surrounding it, I suggested using a boot CD, which would alleviate all of their security concerns. They gave me the go ahead to give it a try. I now have a working prototype and this document describes the process used to create it.

1.1 Copyright

1.2 Disclaimer

Use the information in this document at your own risk. I disavow any potential liability for the contents of this document. Use of the concepts, examples, and/or other content of this document is entirely at your own risk.

All copyrights are owned by their owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3 News

- V0.06 2007−02−09 (pre−release)
- V0.05 2007−02−05 (pre−release)
- V0.04 2007−02−05 (pre−release)
- V0.03 2006−12−12 (pre−release)
- V0.02 2006.12.11 First Draft (pre−release)
- V0.01 2006.11.13 webgenerated sgml form (pre−release)

1.4 Credits

I used many resources to do what I did. A big thank you to all those that have contributed to these projects. They include but are not limited to:

- http://www.linuxjournal.com/article/7246
- http://www.openvpn.net/howto.html
- http://www.damnsmalllinux.org
- http://www.ubuntu.com

1.5 Translations

Currently there are no translations. If you would you like to translate this, Please contact me at jefferydouglaswaddell (at) gmail (dot) com .

---

2. Theory

To begin, I will attempt to lay out the problem a little more thoroughly and give the theory behind the prototype.

In the case of the Institution, there is a set of machines that are behind a very robust firewall. This firewall allows VERY little to go through. As far as connecting out to the Internet, many things, including access to HTTP, FTP, secure shell, etc., are allowed. As far as connecting in, none of those are allowed.

You can imagine the firewall as a set of machines on the edge of our internal network. There is a lot of unsecured traffic within our internal network, which has both private networks and internet addressed clients within it. To be allowed to do any of the things that require a connection INTO the network from the Internet, a server allowing the connection must be on the periphery of the network (i.e. part of the firewall).

The desire has been to be able to access the internal network sufficiently to do work from home (on par with doing work in the office), but without opening up the firewall such that our unsecured traffic would be exposed to the Internet. There is also sensitive, internal data within the internal network that can be accessed, that should neither be copied to the home machine (laptop, etc.) nor printed at the home machine.

So the question becomes: how do you allow enough access to give certain people the ability to do their jobs from home and still protect the sensitive information and the network as a whole? Other security considerations that came up were:

- The potential for a laptop with sensitive information on it to be stolen.
- A home machine to being used in a way that breaks policy and yet can't be traced or effectively enforced due to it's off−site actuality.

The answer that came to my mind was to give the users a boot CD that they could place in a machine (whether at Uncle Joe's, their own house, or the internet cafe at the airport) and use to boot into their internal work desktop. The theory is that connecting to a server on the periphery of the network, which then gives them access to their desktop, is at least as secure as if they had accessed the network from their internal desktop, and that a great deal of control can be exercised over when and how that access is granted. Being a boot CD, no on−site OS configuration will be required, nor will extra precautions concerning malware, spy−ware, or viruses need to be taken. In the next section, Technologies, I will examine some of the ways this can be implemented, and in the Implementation section I will explain how we actually did it.

3. Technologies

Since many people are already familiar with openVPN, this seemed like a good idea. However, in and of itself openVPN is not sufficient. The most convenient way for people to be able to work is for them to be able to directly connect to their already existing desktop. All of the users here run either Windows XP (tm) or Windows 2000 (tm), which suggests rdesktop as a solution. But rdesktop can't get through the firewall and we won't open our firewall for that traffic as it would be too hard to secure. Adding openVPN allows for more security, but it runs into the following problems within the scope of our implementation:

1. We would have to set up openVPN server on EACH internal desktop and each external client machine (at home, at the Internet cafe, or wherever) and it would therefore be highly subject to IT time and in general be a pain to get set up and working for each separate user's setup.
2. If you allow a direct remote session (even over openVPN) you run into several potential security risks.
    1. key loggers on the external client box
    2. attacks directly on the internal Windows(tm) box through the VPN ports that are now open and exposed on the Internet.
    3. viruses, spy ware and other malware on the client box infecting the internal workplace desktop (and any others that it has connection with) through the established VPN connection.
    4. having the private key stored on multiple desktops around the organization on unsecured desktops. Someone with access to that key (which would need to be on the internal machine in order to establish the the VPN connection) could allow unauthorized key−making.
3. Only the specific external machine that is setup by IT services personnel would be able to connect and use the resources, when what is actually desired is that the authorized user can get access from anywhere.

In order to eliminate the security issues above and to make it less of a difficult system to maintain into the future, I suggested creating a Linux live CD that boots, logs into an openVPN server that connects the external and internal networks, and then automatically opens the individuals internal desktop using rdesktop.

4. Implementation

Many decisions had to be made as to which direction to go. The following subsections detail some of the paths I took to get to a working prototype. Please modify to suit your environment. Where appropriate I will make clarifying comments.

I looked at several live CD distributions and concluded that DSL would work best for the purposes at hand. I considered the following:

1. Ubuntu. This live CD expects a relatively high−end machine and has way more applications than would be useful to this project. Instead of trying to remove tons of applications and whittle it down I opted not to use this one. It could be a good choice though if you are trying to give people a full desktop PLUS access to an internal network.
2. PuppyLinux. This live CD looked really good, however I had trouble figuring out the SFS file system it uses for its root, and was able to get to instructions and tools on how to deal with the Knoppix compressed file system much more easily.
3. Knoppix. This live CD, like Ubuntu, was too top heavy for the specific purposes of this project.
4. Damn Small Linux. This live CD has a 50 MB footprint, will work on almost anything hardware−wise, and is what I chose to implement.

4.1 Getting and modifying a DSL−based .ISO file.

The choice of DSL means that we are relying on DSL's built in ability to automatically find, configure and attach to a network via DHCP. DSL's wireless support is very minimal and thus we do not support wireless at this time. The end user will need a machine that normally attachs to the Internet through DHCP from their provider and uses a normal wired network card to do so.

1. Fresh install of Ubuntu (http://www.ubuntu.com), EdUbuntu (http://www.edubuntu.org), xUbuntu (http://www.xubuntu.org), or kUbuntu (http://www.kubuntu.org)
2. Use Synaptic to add repositories (all available)
3. Install qemu, open−vpn and cloop−utils
4. Get an ISO (I recommend the dsl−3.0 ISO); hopefully I'll soon have one of my prototype CD images (sans VPN keys) available on the Internet for your downloading pleasure. Refer the Examples section − this ISO might be a good place for you to start.
5. Mount the ISO somewhere.
    1. `mkdir /tmp/workingiso`
    2. `mount -t iso9660 -o loop dsl-3.0.iso /tmp/working.iso`
6. Unpack the compressed file system of the ISO
    1. `extract_compressed_fs /tmp/workingiso/KNOPPIX/KNOPPIX > /var/tmp/KNOPPIX-cloop`
    2. Mount it somewhere
        1. `mkdir /tmp/workingiso.cloop`
        2. `mount -o loop /var/tmp/KNOPPIX-cloop /tmp/workingiso.cloop`
7. Now that you have access to the inner workings of the CD, copy that to a place where you can work with it.
    1. Make a directory to work in (i.e. /home/jeff/Desktop/vpn−tree)
    2. `tar -C /tmp/workingiso.cloop -cf - . | tar -C /home/jeff/Desktop/vpn-tree -xvpf -`
8. Also copy the outer part of the CD, where you can work with it.
    1. Make a directory to work in (e.g. /home/jeff/Desktop/vpn−cd−tree)
    2. `tar -C /tmp/workingiso -cf - . | tar -C /home/jeff/Desktop/vpn-cd-tree -xvpf -`
9. Make a CD image with what you have now to confirm you've made it this far without error.
    1. `mkisofs -pad -l -r -J -V "YOURVPN v0.1" -no-emul-boot -boot-load-size 4 -boot-info-table -b boot/isolinux/isolinux.bin -c boot/isolinux/boot.cat -hide-rr-moved  -o yourvpn.iso /home/jeff/Desktop/vpn-cd-tree/`
10. Assuming the above worked you can now test it with
    1. `qemu -boot d -cdrom yourvpn.iso`
11. Now you can start making changes.
    1. Mount your proc using

```
mount -t proc none /home/jeff/Desktop/vpn-tree/proc
```
2.
```
chroot /home/jeff/Desktop/vpn-tree
```
3. Make any changes you would like to the file system.

12. After messing around, it's time to write out your new compressed file image and make a CD.
    1. Exit from chroot
    2. Unmount the image's proc (don't forget this step or you will not have a working image when you build it later)
    3. Make the compressed file image

    ```
    mkisofs -L -R -l -V "YOURVPN ISO9660" -v -allow-multidot /home/jeff/Desktop/vpn-tree/ | create_compressed_fs - 65536 > /home/jeff/Desktop/vpn-cd-tree/KNOPPIX/KNOPPIX
    ```
    4. Make the cd image

    ```
    mkisofs -pad -l -r -J -V "YOURVPN v0.2" -no-emul-boot -boot-load-size 4 -boot-info-table -b boot/isolinux/isolinux.bin -c boot/isolinux/boot.cat -hide-rr-moved -o yourvpn.iso /home/jeff/Desktop/vpn-cd-tree/
    ```
    5. Test it in an emulator (I tend to enjoy qemu...use whatever you like: vmware, xen, ?)

    ```
    qemu -boot d -cdrom yourvpn.iso
    ```
13. Repeat as necessary to get the desired ISO image.
14. Burn image and enjoy.

4.2 Comments on modifying the software on the CD.

1. Unpack openvpn*.deb to the root file system after chroot.
2. Make sure all the proper libraries were copied to the proper place.
    1. chroot
    2. ldd /usr/sbin/openvpn
    3. Go to the other root terminal on the main system and copy any libraries from the main system to the vpn−tree
    4. Make the tun node: mknod /dev/net/tun c 10 200
3. Remove the loading of the DSL business card graphic by editing the vpn−cd−tree/boot/isolinux/boot.msg file and removing "^Xlogo.16"
4. Cause it not to wait for boot options by editing vpn−cd−tree/boot/isolinux/isolinux.cfg and changing the line that says "PROMPT 1" to "PROMPT 0"
5. Edit the file vpn−tree/etc/skel/.xinitrc to reflect what we wish to happen on the desktop. Remove code to load icons onto desktop; remove code to make the windows see through; add code to establish VPN connection; add code to load rdesktop and make connection to correct machine.
6. Edit the display screen for boot.
    1. copy vpn−cd−tree/boot/isolinux/minirt24.gz to /tmp
    2. gunzip minirt24.gz
    3. mount −o loop minirt24 /mnt
    4. edit /etc/linuxrc to display text indicating Institution's name (you would put whatever is appropriate for your institution here) instead of "DSL"
    5. umount /mnt
    6. gzip minirt24
    7. copy minirt24.gz over to vpn−cd−tree/boot/isolinux/minirt24.gz

4.3 Setting up multi user OpenVPN server.

1. Follow instructions on making cert and keys for the server.
2. You will need to enter several pieces of information that are covered in the openVPN HOWTO.
3. Remember to create a password−protected key for the client.
4. Set all the configuration as desired.

5. For each client you will need to
    1. make a password−protected key using the certificate
    2. place the certificate, and client key (only) in the vpn−tree/etc/openvpn/keys directory
    3. adjust the vpn−tree/etc/openvpn/openvpn.cfg file to have the proper key files indicated (see the server configuration file in the Samples section)
    4. adjust the added routes in the vpn−tree/opt/bootlocal.sh
    5. adjust the vpn−tree/etc/skel/.xinitrc to point to the correct rdesktop IP.
6. Rebuild the CD.
7. Test in the emulator.
8. Once it works correctly, either burn the ISO or make a qemu Windows emulator version by placing the .ISO in the win−qemu−yourvpn−cd directory and building that .ISO (don't forget to burn it afterwards).

4.4 References:

1. http://www.linuxjournal.com/article/7246
2. http://openvpn.net/howto.html
3. http://www.damnsmalllinux.org
4. http://www.ubuntu.com

5. Maintenance

Once built there is NO maintenance to the CDs. If you need to change the private key password for the the individual user, burn them a new CD. If they lose a CD, give them a fresh burn. If the CD gets destroyed, give them a new copy.

The openVPN server requires little maintenance. It is recommended that you periodically check the openVPN logs on the server to determine the likelihood of nefarious activity and act accordingly. Usage−tracking is beyond the scope of this document.

6. Advanced Issues

I believe that the majority of security and ease of use issues are dealt with by the technologies I have described here. Certainly there are many issues that this technology could expose (mostly internal political or policy issues within your own institution).

If you discover any flaws in the technology. Please contact me about it.

Also, if you can explain the proper pf rules to get *bsd to properly forward (masquerade the 10. network) packets to Internet−routed network segments on the internal side of the VPN, I would definitely like to hear from you. I could not get it to work, whereas the Linux masquerading rule I found just works.

7. Features

1. Ease of use for the end user:
    1. Put in CD
    2. Boot machine
    3. Type in private key password

    4. Log into work desktop and work as usual
2. Ease of use for the administrator(s):
    1. Key generation is separate from use.
    2. A user's access can be specifically revoked (without affecting their work desktop) using a single command.
    3. All new users can be denied by shutting down the openVPN server process on the server.
    4. All connections can be broken by shutting down the entire server; this will also deny future access until the server is brought back up and the end user reboots.
3. The CD build process can be automated for ease of creation.
4. The openVPN logs can be used to determine (or trace) nefarious or out–of–policy computer use.

8. Troubleshooting

There are several troubleshooting techniques that I used in the course of this project. I would expect people attempting this to be familiar with most or all of them. Here are a few:

1. tcpdump is your friend when attempting to prove that your traffic really is going over the VPN.
2. ldd is very helpful in finding out what libraries are missing when you have to install a package by hand.
3. qemu or another emulator is priceless when you don't wish to waste a bunch of physical CDs or the time it takes to burn and boot them.

9. Further Information

As mentioned elsewhere, please see http://openvpn.net for more information on openVPN...the HOWTO is especially good.

Please see http://damnsmalllinux.org for more info on DSL.

For live CD information, Google may be your best bet, although there is now a live CD book that seems fairly good.

10. Getting Help

Obviously the above mentioned communities can help you with varying aspects of the respective pieces of this puzzle.

If you need help understanding how to put it all together, feel free to contact me after you've thoroughly read (and maybe tried) this HOWTO.

11. Concluding Remarks

There you have it. May it be of use.

12. Questions and Answers

When I receive questions generated from this document I will compile them and insert them here.

---

13. Bits and Pieces

13.1 Making a Windows autorun CD.

Due to a policy decision, we will not be deploying this, although it does work. The security concerns over this method include the following:

1. Key logger on the host Windows (tm) machine. This could conceivably be used to capture the private key password and potentially grant unauthorized access.
2. Malware on the host Windows (tm) machine. Might be able to send through the VPN...seems unlikely.
3. A virus on the host Windows (tm) machine. Might be able to propagate itself through to the internal network...again this seems unlikely.

This is what you do to create one. This method is likely useful for other projects.

1. ```
   mkdir win-qemu-yourvpn-cd
   ```
2. Download qemu−0.8.2−windows.zip from http://www.h7.dion.ne.jp/ qemu−win/
3. Unzip qemu−0.8.2−windows.zip into the win−qemu−yourvpn−cd directory.
4. Move all the qemu−0.8.2−windows files up one directory. Remove the qemu−0.8.2 directory.
5. Make an icon file. I used a stock one and resized with GIMP.
6. Create an autorun.inf file in win−qemu−yourvpn−cd directory containing the following:

   ```
   [autorun]
   icon=youricon.ico
   open=yourvpn.bat
   ```
7. Copy qemu−win.bat to yourvpn.bat.
8. Edit yourvpn.bat replacing the last line in the file with: qemu.exe −L . −m 64 −soundhw all −localtime −cdrom yourvpn.iso
9. Copy the fully made bootable .ISO image yourvpn.iso from where it is currently to win−qemu−yourvpn−cd
10. Make an ISO of this directory: mkisofs −pad −l −r −J −V "WQYOURVPN v0.1" −hide−rr−moved −o wqyourvpn.iso /home/jeff/Desktop/win−qemu−yourvpn−cd/
11. Burn the ISO and try it on a Windows (tm) box.

---

14. Examples

Here is the config file (with IP address and key names removed for the CD). Text in upper case is used to indicate that you need to change whatever is there to your setup.

−−−−−−−−−−−−−−−−−

```
##############################################
# 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.
;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.
;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 PUT_YOUR_OPENVPN_SERVER_IP_HERE 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 nogroup

# 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 /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/THECDUSERSCRTFILE.crt
key /etc/openvpn/keys/THECDUSERSPASSWORDPROTECTEDPUBLICKEY.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

#added in hopes of removing a command line option
auth-retry interact
```

Here is the openVPN configuration file for the server.

____

```
#################################################
# Sample OpenVPN 2.0 config file for            #
# multi-client server.                          #
#                                               #
# Edited on 9/12/2006 by Jeff Waddell           #
#                                               #
# 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 preceded with '#' or ';'         #
#################################################

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

# 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.
;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 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 /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/server.crt
key /etc/openvpn/keys/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 /etc/openvpn/keys/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. See the man page for more info.
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 10.8.0.4/255.255.255.0.  Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients.  Leave this line commented
# out unless you are ethernet bridging.
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100
```

```
# 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 192.168.10.0 255.255.255.0"
;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 60

# 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.
# If you enable it here, you must also
# enable 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 nogroup

# 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 /var/log/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  /var/log/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
```

Still looking for Web space to store the .ISO file.