initial notes on OpenVPN

you can find latest information at the web site:
openvpn.net/

With version 2.0.x OpenVPN is much more useful, scalable, and easier to configure. There are even GUI versions for Linux and M$ Windows ; the Mac OS X GUI - well, it was under development, but opinions vary on what is happening.

the HOWTO is useful, but there are quite a few configuration options.

with Debian, it is just *apt-get install openvpn*

when first tyring it out, using a static key is the easiest, but least safe.

OpenVPN can be used in two device modes: *tun_ or tap . tun basically requires routing if you want a 'client' machine to appear 'inside'. tap is used with bridging. that can be very useful if you want traffic, broadcasts, appletalk, to be passed thru the vpn. tun cannot do that, as it routes, and such traffic is not routable (in a conventional low-level approach...)*

for initial testing, a suggested sequence is to do this: (as much a confidence-building exercise as anything else, but if you encounter problems, you get those out of the way before moving to more complicated but useful configurations)

1. first set up a peer-to-peer vpn with a static key
2. set up a client-server vpn with static key
3. set up a multi-client-server vpn with static keys
4. as above, with individual certs for each client.

The examples below will assume use of debian (sarge) and OpenVPN v2. It is possible to make an OpenVPN v1 client talk to a v2 server, but that will not be discussed here. See the HOWTO (openvpn.net/howto.html), FAQ (openvpn.net/faq.html) and troubleshooting for more info.

a basic server configuration (in /etc/openvpn):
note that in this case, the client configuration is the same except for changing the order of the IP addresses in the ifconfig line, and "up ./server.up" would be instead "up ./client.up", also in the client config you need to have "remote the.remote.hostname" you are going to connect to.

```
dev tun
ifconfig 192.168.100.1 192.168.100.2

# script to run to establish routes
up ./server.up

# Our pre-shared static key
secret static.key
port 1194
user nobody
group nogroup
```

```
# Send a UDP ping to remote once
# every 15 seconds to keep
# stateful firewall connection
# alive.  Uncomment this
# out if you are using a stateful
# firewall.
ping 15

# Verbosity level.
# 0 -- quiet except for fatal errors.
# 1 -- mostly quiet, but display non-fatal network errors.
# 3 -- medium output, good for normal operation.
# 9 -- verbose, good for troubleshooting
verb 3

log-append  /var/log/openvpn/openvpn.log
status /var/log/openvpn/status.log
```

server.up

```
#!/bin/sh
route add -net 192.168.100.0 netmask 255.255.255.0 gw $5
```

client.up

```
#!/bin/sh
route add -net 192.168.100.0 netmask 255.255.255.0 gw $5
```

kinda similar, no?

```
# mkdir /var/log/openvpn
```

but wait there's more !

well, this gets you an encrypted tunnel once you've generated the secret.key
frodo@macaw:> openvpn --genkey --secret static.key

but, you have to ensure that your firewalls (1) allow port 1194 udp on the server and also you will be wanting to add the following rules:

iptables -A INPUT -i tun+ -j ACCEPT
iptables -A FORWARD -i tun+ -j ACCEPT

if you want to do any *routing* on one end, you will need to

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

and you will need to do something like this:

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

the client-server paradigm

What makes OpenVPN v2 much better than v 1.x is that it can have multiple clients for a single server - version 1 requires a separate port per client, if i recall correctly).

In addition, version 2 allows multiple types of authentication. If pam is employed on the server, you can use whatever user authentication scheme you want in pam; or you can use x509 certs and your favorite certificate authority.

You can also have the server hand out IP addresses to the client, so clients can be mobile.

certificates made easy

OpenVPN v2 has a number of scripts to make it easy to set up the certificates used. Deb puts them in /usr/share/doc/openvpn/examples/easy-rsa

the bare-bones. edit the file "vars", source it, and then run the scripts:

./build-ca
./build-key server
./build-key client1
./build-key client2
./build-dh

build-ca creates the ca crt which server and client will both need.
build-key creates keys, which you can sign with the ca.crt
build-key server creates server.crt server.csr and server.key ~ build-key client creates client.crt client.csr and client.key
etc.

build-dh builds a Diffie-Hellman pem file, 1024 bits by default, but you can use 2048 if you want. Only the server needs this

In the examples below, there is also a tls-auth key, created by doing this:
openvpn --genkey --secret ta.key
both server and client would need this.

server and client config examples

An example server.conf file;

```
dev     tun
mode    server
tls-server
# with tls-auth  server is value 0 and client is value 1
tls-auth keys/ta.key 0
dh      keys/dh2048.pem
ca      keys/ca.crt
cert    keys/server.crt
key     keys/server.key
duplicate-cn


server 192.168.100.0 255.255.255.0 # IP range clients
ifconfig-pool-persist ipp.txt
# note: initial tests used these, and they worked, but
# the man page hade the two lines above.
#ifconfig 192.168.100.1 192.168.100.2
#ifconfig-pool 192.168.100.5 192.168.100.200 # IP range clients

route-up "route delete -net 192.168.100.0/24"
route-up "route add -net 192.168.100.0/24 tun0"

push "route 192.168.100.1" # add route to protected network

# the next line tells the client to route all traffic thru the VPN
# you might not want this
```

```
push "redirect-gateway def1"

# if you do not want to route all client traffic thru VPN, do something like
# the following (uncomment out and edit as needed)
#push "route 10.90.134.0 255.255.255.0"
#push "route 10.0.134.0 255.255.255.0"
#push "route 195.214.241.0 255.255.255.0"

# if you have mobile users, the following can be used:
push "dhcp-option DOMAIN riseup.net"  #push the DNS domain suffix
push "dhcp-option DNS 69.90.134.134 "  #push DNS entries to client
push "dhcp-option WINS 69.90.134.134 "  #push WINS entries to client

port 1194

user nobody
group nogroup

; comp-lzo

ping 60
; ping-restart 45
; ping-timer-rem
persist-tun
persist-key

verb 3
log-append      /var/log/openvpn/openvpn.log
status          /var/log/openvpn/status.log

# uncomment the following lines if you want to use PAM but
# note that on debian, you need to apt-get install libpam0g-dev
#plugin /usr/lib/openvpn/openvpn-auth-pam.so common-auth
#client-cert-not-required
```

client.conf

```
dev tun
tls-client
# 1 below means "client"
tls-auth keys/ta.key 1
ca      keys/ca.crt
cert    keys/client1.crt
key     keys/client1.key

# Our OpenVPN peer is the office gateway.
remote stork.riseup.net

pull

;port 1194

user nobody
group nogroup

; comp-lzo
; ping 15
; ping-restart 45
; ping-timer-rem
```

```
;persist-tun
;persist-key

verb 3
log-append      /var/log/openvpn/openvpn.log
status          /var/log/openvpn/status.log

# uncomment the following if the server uses PAM
#auth-user-pass
```

assigning static IP's to 'clients'

from the HOWTO, 'policy' section, as well as in server.conf example, there are a few things you need to do:

1. set up unique certs for each client, and note the CN of the cert.
2. to the server.conf file, add directive:
3. create the ccd directory, and populate it with a file for each client, using the CN for the file name, containing something like the following:

ifconfig-push 192.168.200.1 192.168.200.9

put certs on appropriate clients.

example server.conf

```
dev     tun0
proto   udp
tls-server

# with tls-auth  server is value 0 and client is value 1
tls-auth /etc/certs/ta.key 0
dh      /etc/openvpn/keys/dh2048.pem
ca      /etc/certs/roots/cacert-root.pem
cert    /etc/certs/emu.riseup.net/cert.pem
key     /etc/certs/emu.riseup.net/key.pem

client-config-dir /etc/openvpn/ccd

server 10.8.0.0 255.255.255.0 # IP range clients

client-to-client

port 1194

user nobody
group nogroup

comp-lzo

keepalive 10 120
persist-tun
persist-key

verb 3
log-append      /var/log/openvpn/openvpn.log
status          /var/log/openvpn/status.log
```

directory /etc/openvpn/ccd has files named with the CN of the cert for each client, e.g. for

gull.riseup.net, create a file gull.riseup.net, containing something like this:

```
#ifconfig-push clientIP serverIP
ifconfig-push 10.8.0.3 10.8.0.1
```