OPENVPN - Shared Private Key Setup (Part 1)
AUTHOR - Nic Maurel

Recently I was asked to build a VPN. It's not everyday you get asked to build a VPN. So I went looking for a solution that would best suit my needs and the needs of the users. Straight away I thought of IPsec Tunneling. But then I thought, "IPsec can't be Natted", (Packet changed from a public IP to a Private IP or vice versa), so that blew that idea out the water. I then saw the magic of OpenVPN, an open source SSL based VPN solution, and I grabbed at it. You can check it out at http://openvpn.net. Why did I choose it? This is why...

- SSL tunneling between Linux & Linux (tested on redhat 9)
- SSL tunneling between Linux & Windows
- SSL tunneling between Windows & Windows
- Use static, pre-shared keys or TLS-based dynamic key exchange
- Batch SSL scripts to easily create your own CA and sign certificates
- LZO compression for slow lines
- Choose TCP or UDP protocol
- OpenVPN defaults to 1194 but you can specify any port
- Tunnel networks over NAT
- A Gui or CLI environment for windows (tested on XPSP2 and 2000Pro)
- Integrate iptables into tunnelling forwarding packets
- View Packets being passed (with Ipsec this is not straight forward)
- Bridged or Routed Environment

Now we are talking!
When I first looked at OpenVPN I thought this is going to be tough, but it was actually very simple to setup. I managed to get a share key(static) key working within 30 minutes. Lets take a look at how..

What do you need?

- A Linux Box with 1 or 2 NIC's depending on whether you are putting your vpn server on the same or separate subnet
- OpenSSL installed
- LZO-1.08 installed
- OpenVPN 2.0 installed

I will not focus on installing as there is plenty of documentation on installing OpenVPN on the website. OpenVPN uses a Tun/Tap Adapter. This is a virtual adapter that encrypts and decrypts packets. This is a very undetailed path of the packets coming in and out of a one side of a tunnel, expect a similar thing on the other side.
 Application (eg. Outlook) --> lo 127.0.0.1 --> Tun/Tap - 10.0.0.1 (encrypt) --> Network Adapter 192.168.0.1 -->
 Application (eg. Outlook)<-- lo 127.0.0.1 <-- Tun/Tap - 10.0.0.1 (decrypt) <-- Network Adapter 192.168.0.1 <--

Lets have a look at a basic point to point tunnelling Scenario (Please note the tun ip does not have to 10.0.0.x it can be any private subnet you please)..

```
              OpenVPN Server 1     Firewall                        Firewall      OpenVPN Server 2

              ___________________                                                ___________________
            |                   |   Public IP------- TUNNEL ---------Public IP  |                   |
            | NIC - 192.168.0.1 |>196.x.x.x > ADSL > Internet- ADSL >196.x.x.x->| NIC -192.168.0.1  |
192.168.0.x-| TUN - 10.0.0.1    |                                               | TUN - 10.0.0.2    |
subnet      |___________________|                                               |___________________|
```

To anybody on the internet the packets will just have a source of 196.x.x.x and destination of 196.x.x.x, while to the VPN server 1, packets coming from server 2 have a source of 10.0.0.2 and vice versa. Now that we have an idea of how it all works, we have one more hurdle to jump over before getting down to the nitty gritty.

Bridged or Routed?
This is probably the question people would get most stuck on. You need to cater for your own needs in your specific network scenario. Bridging is easier than routing as one does not need to deal with complex routes as all nodes need to have a route back to the VPN server, but in the end routing is a more secure solution as broadcasting packets will go over the bridged VPN. I would recommend routing unless you have a specific need for bridging. In the next examples I will be showing you routed VPNs. Bear in mind that if you choose bridging you *must* specify the Tap adapter in your configuration file, while if you choose Routing you *can* specify the Tap or Tun adapter (preferably tun) in your configuration file.

Scenarios
There are two scenarios I can think of with a routed network and OpenVPN. With both scenarios you only need one network card if both your firewall and vpn supports ip forwarding. To turn on ip forwarding type this as root on linux:

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

1) Scenario - DMZ
This is the secure way of doing things as there are never routes to internal. All servers are replicated periodically through the vpn eg. File Servers

```
                 Firewall       VPN Server      VPN Subnet
                    _           ___________     ___________

 INTERNET ---- |*|  ---    | 192.168.1.1  |---|192.168.1.x |
                           |___________|    |_ _________|
                    |
                    |
                   \/
               ___________
              | 192.168.0.x |
              -----------------
             Internal Network
```

2) Scenario - Internal
This setup is more insecure as one is accessing internal fileservers directly. All access is locked down to the bare minimum. I would use SSL encryption and make sure you only give the access needed. eg.

```
                 Firewall        VPN Server
                    _           ____________

 INTERNET ---- |*|  ---    | 192.168.1.12 |
                   |        |____________|
                   |
                   \/
               ____________
              | 192.168.0.x  |
              -----------------
             Internal Network
```

There are many other ways to go about it as every network is different, you must choose the best for you.

Configuring OpenVPN to work
Getting OpenVPN to work is so simple. There is one config file per VPN server or client. The configuration can consist of a few lines or up to a 2 pages, this just depends on how robust and secure you want your VPN to be

Setting up Static Key Configuration
This is a very simple setup, as it doesn't need a Certificate Authority to sign certificates for ssl. Basically in this setup you would have two machines with OpenVPN installed. With the static key configuration one can only have one client per server as opposed to multiple clients..

Server 1 - 192.168.0.1
Client 2 - 192.168.0.2

..connected with a crossover,hub or switch.

Once OpenVPN is installed all you need to do for simple encryption is to create a shared secret key for both parties. I will assume the default directory, if you have specified otherwise you would need to change the parameters

```
[root@server1 openvpn-2.0] /usr/local/openvpn-2.0/openvpn --genkey --secret key.txt
```

This will generate a 2048bit encrypted key called *key.txt*, this needs to be copied over to the other OpenVPN server or client with *scp* or *sftp* (don't use *ftp*).

Check if the tun module is loaded..

```
[root@server1 openvpn-2.0]# lsmod | grep tun
                        tun              5696  0  (autoclean)
```

..if it doesn't show up then just type ...

```
[root@server1 openvpn-2.0]# modprobe tun
```

Next we setup configuration files, here are two very basic configuration files, you may modify them to your hearts content.
Please note I would only recommend using these as starter templates..

```
Server 1 - linux - 192.168.0.1 - config.ovpn

 remote 192.168.0.2                          <- remote PC (optional) if left out will accept from any ip
 dev tun                                     <- Device type this can be tun or tap
 ifconfig 10.3.0.1 10.3.0.2                  <- Point to Point route the virtual ip must take
 secret \usr\local\openvpn-2.0\key.txt       <- Specify where you keeping your key
 verb 3                                      <- Verbosity 0 - 9
 mute 10                                     <- Mutes any output after 10 connection attempts
daemon                                       <- Daemon will run in the background.
```

```
Client 2 - Windows - 192.168.0.2 - config.ovpn

remote 192.168.0.1                      <- If you are the client you should specify the server ip
dev tun
ifconfig 10.3.0.2 10.3.0.1              <- Note how the point to point route is reversed
secret C:\\vpn\\keys\\key.txt           <- This is how you specify windows folders
verb 3
mute 10
```

Launch config files from both pc's...

Server 1 /usr/local/openvpn-2.0/openvpn --config config.ovpn
Client 2 - Windows Right Click config.ovpn and select : Start openvpn with this config file

It doesn't matter what order you do it in but you should see an output similar to this on server 1...

```
Fri Jul 29 12:22:58 2005 OpenVPN 2.0 i686-pc-linux [SSL] [LZO] built on Jun 23 2005
Fri Jul 29 12:22:58 2005 IMPORTANT: OpenVPN's default port number is now 1194, based on an official port
number assignment by IANA.  OpenVPN 2.0-beta16 and earlier used 5000 as the default port.
Fri Jul 29 12:22:58 2005 Static Encrypt: Cipher 'BF-CBC' initialized with 128 bit key
Fri Jul 29 12:22:58 2005 Static Encrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
Fri Jul 29 12:22:58 2005 Static Decrypt: Cipher 'BF-CBC' initialized with 128 bit key
Fri Jul 29 12:22:58 2005 Static Decrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
Fri Jul 29 12:22:58 2005 TUN/TAP device tun0 opened
Fri Jul 29 12:22:58 2005 /sbin/ifconfig tun0 10.3.0.1 pointopoint 10.3.0.2 mtu 1500
Fri Jul 29 12:22:58 2005 Data Channel MTU parms [ L:1544 D:1450 EF:44 EB:4 ET:0 EL:0 ]
Fri Jul 29 12:22:58 2005 Local Options hash (VER=V4): 'd6e09596'
Fri Jul 29 12:22:58 2005 Expected Remote Options hash (VER=V4): '09bcadc2'
Fri Jul 29 12:22:58 2005 UDPv4 link local (bound): [undef]:1194
Fri Jul 29 12:22:58 2005 UDPv4 link remote: [undef]
Fri Jul 29 12:24:18 2005 Peer Connection Initiated with 192.168.0.2:1194
Fri Jul 29 12:24:18 2005 Initialization Sequence Completed
```

Now you should be able to ping through the tunnel from both sides. On Server1 open up another session and ping 10.3.0.2 ...

```
[root@server1 root]# ping 10.3.0.2
PING 10.3.0.2 (10.3.0.2) 56(84) bytes of data.
64 bytes from 10.3.0.2: icmp_seq=1 ttl=128 time=7.58 ms
64 bytes from 10.3.0.2: icmp_seq=2 ttl=128 time=0.953 ms
64 bytes from 10.3.0.2: icmp_seq=3 ttl=128 time=0.967 ms
64 bytes from 10.3.0.2: icmp_seq=4 ttl=128 time=0.976 ms

--- 10.3.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3036ms
rtt min/avg/max/mdev = 0.953/2.620/7.587/2.867 ms
```

And from Client2 (the Windows machine)...

```
Pinging 10.3.0.1 with 32 bytes of data:
Reply from 10.3.0.1: bytes=32 time<1ms TTL=64
Reply from 10.3.0.1: bytes=32 time<1ms TTL=64
Reply from 10.3.0.1: bytes=32 time<1ms TTL=64
Reply from 10.3.0.1: bytes=32 time<1ms TTL=64

Ping statistics for 10.3.0.1:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds:
  Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Now you can forward all your packets through this tunnel and it will be encrypted. YAY! Pretty useful for Wireless networks, public networks, etc.

Conclusion
I hope you have enjoyed and learned a bit about how OpenVPN works. In Part 2, I will be showing, you how to harden the connection with robustness and SSL encryption as well as adding logging. We will also be integrating natting and iptables into our scenario. Have fun playing with OpenVPN.