Setting up a Layer 3 tunneling VPN with using OpenSSH

Posted by emeitner on Mon 2 Jul 2007 at 16:37

This article describes how to use the new tunneling features of OpenSSH V 4.3 to establish a VPN between two Debian or Debian-like systems. Note that by tunneling I am referring to layer-3 IP-in-SSH tunneling, not the TCP connection forwarding that most people refer to as tunneling.

When operational this VPN will allow you to route traffic from one computer to another network via an SSH connection.

This is a brief recipe rather than a "HOW-TO". It it assumed you are familiar with all of the basic concepts.

Requirements

- Debian Etch and/or Ubuntu Edgy systems
- SSH version 4.3 or higher is required on both ends of the VPN.

Introduction

SSH V 4.3 introduced true layer-2 and layer-3 tunneling allowing easy to configure VPNs that can be built upon existing SSH authentication mechanisms. The VPN configuration described below allows a client(or if you prefer the stupid term: *road warrior*) to connect to a firewall/server and access the entire private network that is behind it.

Previously I never allowed root login via SSH to any machines because I always logged in under a personal account and then used *sudo*. It made sense to turn off root logins via SSH(PermitRootLogin=no). With the advent of the new tunneling features there seems to be a need to have a limited root login for the purposes of establishing the SSH VPN. This is required because the user that connects to the sshd server must have the permissions to set up a tunnnel(tun) interface. Until OpenSSH allows non-root users to do so (such as: http://marc.info/?l=openssh-unix-dev&m=115651728700190&w=2) we will have to do it this way.

OpenSSH also has a few features to allow for easily tearing down an SSH connection without having to track all sorts of PIDs and such. I use the *control connection* feature to do so. See the SSH man page for these switches: -M -O -S. This allows one to use the *ifup* and *ifdown* commands to easily control the SSH VPN.

Scenario

In this recipe two machines will be configured:

- A server which is a firewall and has access to a private network ¹
- A client which initiates the connections to the server and gains direct access to the private network

```
 --------          /\_/-\/\/-\        -----------------
| Client |~~~~~~~/ Internet /~~~~~~| Server/Firewall |~~~[ private net ]
 --------        \_/-\/\_/\/      / ----------------- \
```

```
    ||\                          \         ||\       \
    || {tun0}                     {eth0}   || {tun0}  {eth1}
    ||                                     ||
    \-================ tunnel =============-/
```

For this recipe lets number things like this:

- the private net is 10.99.99.0/24
- eth0 on the server has public IP 5.6.7.8
- eth1 on the server has private IP 10.99.99.1
- the VPN network is 10.254.254.0/30
- tun0 on the server has private IP 10.254.254.1
- tun0 on the client has private IP 10.254.254.2

On the Client

If you do not already have them, generate an SSH keypair for root:

```
$ sudo ssh-keygen -t rsa
```

/etc/network/interfaces: Add this stanza to the file:

```
iface tun0 inet static
      pre-up ssh -S /var/run/ssh-myvpn-tunnel-control -M -f -w 0:0 5.6.7.8 true
      pre-up sleep 5
      address 10.254.254.2
      pointopoint 10.254.254.1
      netmask 255.255.255.252
      up route add -net 10.99.99.0 netmask 255.255.255.0 gw 10.254.254.1 tun0
      post-down ssh -S /var/run/ssh-myvpn-tunnel-control -O exit 5.6.7.8
```

The first time we connect to the server as root we may need to acknowledge saving the servers SSH key fingerprint:

```
$ sudo ssh 5.6.7.8
The authenticity of host '5.6.7.8 (5.6.7.8)' can't be established.
RSA key fingerprint is aa:fe:a0:38:7d:11:78:60:01:b0:80:78:90:ab:6a:d2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '5.6.7.8' (RSA) to the list of known hosts.
```

Don't bother logging in, just hit CTRL-C.

On the server

/etc/ssh/sshd_config: Add/modify the two keywords to have the same values as below.

```
PermitTunnel point-to-point
PermitRootLogin forced-commands-only
```

The *PermitRootLogin* line is changed from the default of **`no`**. You do restrict root SSH login, right?

/root/.ssh/authorized_keys: Add the following line.

```
tunnel="0",command="/sbin/ifdown tun0;/sbin/ifup tun0" ssh-rsa AAAA ..snipped.. == root@server
```

Replace everything starting with "ssh-rsa" with the contents of root's public SSH key from the client(/root/.ssh/id_rsa.pub on the client).

/etc/network/interfaces: Add the following stanza.

```
iface tun0 inet static
      address 10.254.254.1
      netmask 255.255.255.252
      pointopoint 10.254.254.2
```

/etc/sysctl.conf: Make sure `net.ipv4.conf.default.forwarding` is set to `1`

```
net.ipv4.conf.default.forwarding=1
```

This will take effect upon the next reboot so make it active now:

```
$ sudo sysctl net.ipv4.conf.default.forwarding=1
```

Using the VPN

```
user@client:~$ sudo ifup tun0
RTNETLINK answers: File exists
run-parts: /etc/network/if-up.d/avahi-autoipd exited with return code 2

user@client:~$ ping -c 2 10.99.99.1
PING 10.99.99.1 (10.99.99.1) 56(84) bytes of data.
64 bytes from 10.99.99.1 icmp_seq=1 ttl=64 time=96.3 ms
64 bytes from 10.99.99.1 icmp_seq=2 ttl=64 time=94.9 ms

--- 10.99.99.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 94.954/95.670/96.387/0.780 ms
user@client:~$ sudo ifdown tun0
Exit request sent.
```

You may get the two errors after running *ifup*. No problem, they are harmless.

Things to watch for

- Client side VPNs. Firewalls such as Firestarter will block all traffic to and from any tun interface. You will need to modify the scripts in /etc/firestarter to get around this.

Next steps

Once you have this running it is fairly easy to route traffic between two networks on each end of the VPN. See the first reference link below for details.

Possible improvements

- Something to monitor and restart the VPN if it fails, such as autossh: http://packages.debian.org/stable/net/autossh
- Automatic starting of the VPN upon first packet from client destined for the remote private network.
- Use of a remote DNS server by client when VPN is active.

References

- https://help.ubuntu.com/community/SSH_VPN
- http://wouter.horre.be/node/63
- `man ssh`
- `man ssh_config`
- `man sshd_config`
- `man interfaces`

1) The server **can** be behind a firewall, but this requires some additional configuration of the firewall. Primarily, the firewall must forward some port to port 22 on the server. The firewall will need to also know how to route traffic destined for the VPN to the server.