

# OpenSSH VPN

Contributed by TuxInvader  
 Saturday, 15 July 2006  
 Last Updated Wednesday, 23 July 2008

OpenSSH 4.2 and above now have true vpn functionality using tun(4) devices. Unfortunately this functionality has not yet made it into the portable version (not in Linux at least). Don't panic though, you don't need tun to use SSH as a VPN UPDATE \* OpenSSH 4.3p1 added support for FreeBSD, NetBSD and Linux tun devices. \* OpenSSH 4.4p1 added support for MacOS/X and Darwin (via 3r party tun driver).

The new tunnel functionality is great, but myself and others have been using SSH as a VPN for years without it. The tunnel device has just made this easier. With almost any version of OpenSSH you can use the first two methods below. If you have version 4.2 on OpenBSD or a recent version of portable (See above) you may be able to use a tunnel device for your VPN.

- Port Forwarding
- Point-to-point (pppd)
- tunnel device

Port Forwarding Port forwarding is your poor mans VPN. It allows you to forward a single port through the SSH tunnel to a host on the other end. This could be useful if you wanted to access an internal webserver, the admin interface on your DSL router for instance. If you have an application that uses a single port that you want to wrap some security around the standard port forwarding can help.

```
ssh -L 8888:127.0.0.1:80 remote.host.com
```

Forward the localport 8888 to the localhost on the remote side port 80

```
ssh -L 1234:192.168.0.10:25 remote.host.com
```

Forward the localport 1234 to host 192.168.0.10 on the remote network port 25 ssh -R 8888:10.10.10.10:80 remote.host.com Forward the remote port 8888 to 10.10.10.10 on the local LAN port 80

Point to Point (PPPD) This one is a little trickier, but it allows you to use both ends as routers and setup a full VPN link between both networks. We'll tunnel ppp through the ssh connection creating a ppp interface on both sides which you can then route through. pppd updetach noauth passive pty "ssh root@<hostname> pppd nodetach notty noauth" <localIP>:<RemotelP> We're giving the local ppp daemon the following options:

- updetach == Once the connection is up detach from the controlling console
- noauth == Do not require the remote side to authenticate (requires root)
- passive == Wait for the other end to start ppp negotiation
- pty "ssh...." == Use the ssh connection for the ppp conversation
- localIP:RemotelP == Assign these IP address to either end of the connection Inside the pty string we're launching ssh and running pppd on the remote host and telling it not to detach from this terminal, not to attempt to open another terminal and to not require authentication. We don't tell the remote side to be passive because as soon as it is launched we want it to start negotiating with our local daemon. If all goes well after a few seconds the local pppd should detach from your terminal and you should have a new ppp\* interface up and active with the <localIP> Tunnel device To enable the tunnel device you need to add a setting to your sshd\_config: PermitTunnel yes If you don't do this you will get the error "sys\_tun\_open: failed to open tunnel control interface: Permission denied". To use the tunnel you just pass the -w flag to ssh. The format is "-w localdevice:remotedevice", so if you wanted to use tun0 on both ends you could use "0:0", if you're not sure and don't want to specify a tunnel device, you can use any ie "-w any:any" ssh root@some.host.com -w any:any That's it, you should have tun{n} device on both sides of the connection. You now need to assign an IP to both ends and add appropriate routes for the traffic. For example: local side ifconfig tun0 172.16.0.1 up route add -net 172.16.0.0/24 tun0 route add -net 192.168.x.y gw 172.16.0.2 remote side ifconfig tun0 172.16.0.2 up route add -net 172.16.0.0/24 tun0 route add -net 192.168.a.b gw 172.16.0.1 This would allow nodes on the local network (192.168.a.b) to communicate with nodes on the remote network (192.168.x.y). Assuming the nodes have a route using the local/remote nodes for those networks.