Covert communications on Linux

SECRET TUNNELS

Moving data to and from Linux systems under the radar. **BY KURT SEIFRIED**

We have all been there: You plug in or connect to the wireless network, and it doesn't work right. Then you try to *ssh* to your server and you get "connection failed." Trying to connect to your mail server on port 25 using TLS (Transport Layer Security, aka encryption) leaves you staring at the banner for the local ISP's mail proxy, or you get another failed connection. But all is not lost – at least you can surf the web. Unfortunately, every time you mistype a URL, you end up at the ISP's search page, and anything with questionable content, such as *hacking*, is blocked.

At this point, you have two choices: find a good book to read, or use VPN software to get a connection to a remote host by bypassing whatever breakage or filtering is occurring. However, some of the really evil – or just plain incompetent – ISPs also block common VPN software and SSH in an effort to prevent unfettered Internet access through their networks.

Much like trying to stop a cat from getting into a cardboard box, if you try to prevent geeks from doing something, the chances are they will only try that much harder.

If you can pass any form of data to a remote system (IPSec, SSH, http, instant messages, smoke signals), then you can use that channel to carry anything you want. An ISP can only block or filter so much traffic before it becomes completely unusable. The trick is to find a network protocol that is allowed and that is not modified (much) on the fly and that can do this with existing software that lets you tunnel data over the top of it.

Fortunately, three basic network protocols – ICMP, DNS, and http – are almost always allowed, as well as a wide variety of other protocols, such as SSH and instant messaging. If you are lucky, you will be able to use software such as SSH with port forwarding or VPN capabilities over an allowed port, such as 80 (http). If you are unlucky, the ISP will force you through web proxies and their own DNS servers in order to access the Internet.

Tunneling via ICMP

ICMP is a great protocol for tunneling data because it's almost always allowed (blocking it breaks a great many things) and can carry a lot of data [1]. An ICMP packet has 20 bytes of data in the header (the usual source, destination, etc.) and 8 bytes of payload data (type of message, code, etc.), plus a variable amount of other data. The amount of other data sent in the ICMP packet is generally only limited by the maximum packet size on a given network (for Ethernet, generally 1,500 bytes), which is usually true for most wireless networks too. This means that you can send a lot of data over ICMP packets with very little overhead.

When it comes to ICMP tunneling software, you have a couple of options, but your best bet is Ping Tunnel (Ptunnel) because it's the most up to date [2]. Installing Ptunnel is relatively straightforward; RPMs for the second latest release are available courtesy of Dag [3].

To install and build the source, enter:

```
rpm -Uvh http://dag.wieers.com/rpm/
packages/ptunnel/
ptunnel-0.61-1.rf.src.rpm
cd /usr/src/redhat/
rpmbuild -ba ptunnel.spec
```

If you want the most recent version of Ping Tunnel, you'll need to update the source RPM or build it from source. To update the source RPM:

```
wget http://www.cs.uit.no/~daniels
/PingTunnel/PingTunnel-0.70.tar.gz
tar -xf PingTunnel-0.70.tar.gz
cd PingTunnel
make
make install
```

Because building it from source is a two-line example, I leave updating the source RPM as an exercise for the reader.

Listing 1: Proxytunnel

```
01 Host proxytunnel.example.org
02              ProtocolKeepAlives 30
03              ProxyCommand /path/to/proxytunnel -p proxy.customer.com:8080 -u
   user -s password -d proxytunnel.example.org:443
```

Running Ptunnel isn't much more difficult. On the server side (the proxy), you simply run Ptunnel with an optional network device (*-c*) and a password (*-x*). On the client side, you specify the address of the proxy server, the local port to listen on, and the remote address and port to which you want to connect. The following example assumes that a proxy server at *ptunnel.example.org* is connected to the Internet via eth0, with a Squid proxy running on the server *squid.example.org* on port 3128 and using the password *blahblah* to secure the connection:

```
Server:
./ptunnel -c eth0 -x blahblah
Client:
./ptunnel -p ptunnel.example.org ➥
-lp 3128 -da squid.example.org ➥
-dp 3128 -x blahblah
```

Now just point your local web browser at localhost port 3128 as a web proxy, and your http traffic will be converted to ICMP traffic and then sent to *ptunnel.example.org*. There, it is unpacked and sent on to the *squid.example.org* web proxy server and then sent to the Internet at large. The squid server can be run locally on the same server running Ptunnel, which will allow you to bypass any filtering and most network breakage completely.

Tunneling via DNS

Although it is not quite as reliable as ICMP, DNS is another protocol that can be used to tunnel data. Some ISPs redirect unregistered or unknown systems to a payment gateway. To do this, they answer any DNS query with the IP address of the payment gateway. Other ISPs will simply use a transparent web proxy to intercept any WWW requests and redirect them to their payment gateway (in this case, you can probably tunnel your traffic over DNS).

DNS offers several advantages over ICMP. Although blocking ICMP does cause problems, it can be done. On the other hand, blocking DNS breaks everything. Although Ping Tunnel 0.70 now supports transmission of data over port 53 UDP, it doesn't actually send valid DNS packets, so you can't pass this traffic through DNS servers. For this, you must have a direct connection to your proxy server, in which case, you can simply use OpenVPN or OpenSSH over port 53.

For an actual proxy that encapsulates the data in valid DNS packets, you have a couple of options: OzymanDNS [4][5] and NSTX [6]. Unfortunately, the NSTX project hasn't updated their source code since 2002, and you will have to use CVS to download it because the source packages seem to be gone. Additionally, because of several design issues, NSTX is quite slow. With no updates since its initial release, OzymanDNS is also somewhat out of date.

Tunneling via Http

Your final option is to tunnel traffic over http or https [7]. Chances are pretty good that any network you are on will allow outgoing https connections. Https is better than http because https encrypts traffic, so the chance that the data will be modified is less. As with the ICMP tunneling software, you can either build Proxytunnel from source or you can download a source (or binary) RPM [8] [9].

To use Proxytunnel, simply run it on your server either as a standalone daemon or from *inetd*, and on the client side, add it as a *ProxyCommand* to your OpenSSH client (Listing 1).

As you can see, are a variety of options are available for tunneling network traffic over networks, depending on protocol availability. A little bit of setup in advance can save you a lot of trouble when you're stuck on someone else's broken, filtered, or otherwise not-working network. ■

SSH Port Forwarding and VPN Capabilities

Virtually all systems have an SSH server, and SSH clients are easy to come by [10]. If you can establish a direct TCP connection to port 80, you can simply run an SSH server on port 80 (*ListenAddress 10.2.3.4:80* in *sshd_config*, but make sure you also define *Port 22* if you also want it to accept connections on port 22 as usual) and enable port forwarding (*AllowTcpForwarding* set to *yes* in *sshd_config*).

```
ssh -L 8080:www.example.org:80 user@
ssh-server
```

Another option with SSH is to use its VPN capabilities. The advantage of this is that you can easily route all your traffic over the connection (Instant Messenger, BitTorrent, etc.) [11].

INFO

[1] Project Loki – ICMP Tunneling: *http://www.phrack.org/issues.html?issue=49&id=6#article*

[2] Ping Tunnel (Ptunnel): *http://www.cs.uit.no/~daniels/PingTunnel/*

[3] Ping Tunnel RPM: *http://dag.wieers.com/rpm/packages/ptunnel/*

[4] OzymanDNS: *http://www.doxpara.com/?p=51*

[5] OzymanDNS HOWTO: *http://www.dnstunnel.de/*

[6] NSTX: *http://savannah.nongnu.org/projects/nstx/*

[7] Tunneling SSH over http(s): *http://dag.wieers.com/howto/ssh-http-tunneling/*

[8] Proxytunnel: *http://proxytunnel.sourceforge.net/*

[9] Proxytunnel RPM: *http://dag.wieers.com/rpm/packages/proxytunnel/*

[10] SSH Port Forwarding article: *http://magazine.redhat.com/2007/11/06/ssh-port-forwarding/*

[11] OpenSSH Layer 3 VPN: *http://www.debian-administration.org/article/Setting_up_a_Layer_3_tunneling_VPN_with_using_OpenSSH*

THE AUTHOR

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.