Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

SSH Port Forwarding

Giles Orr
Jacob Wyatt
Georgia College & State University

SSH Port Forwarding allows the use of the encrypted SSH tunnel between hosts to forward information on connections that would not normally be encrypted. Using this powerful tool is initially daunting, but is fairly straight−forward when it is understood.

1. Introduction

SSH stands for "Secure SHell." SSH is a replacement for telnet, rsh, and rlogin, to allow secure shell access to remote machines over an untrusted network. Telnet was designed at a time when the Internet consisted of a relatively small number of universities, and no one had ever heard of a packet sniffer. Packet sniffers such as sniffit and tcpdump are now relatively common − they have some highly practical uses, but obviously can also be used to collect passwords of those using unencrypted connections on a local network. Even if the password handshaking is encrypted, quite a bit of personal information can be collected watching an unencrypted transaction after the passwords. SSH packets looks like garbage to a packet sniffer.

2. Available Versions

SSH is currently available for free in several different versions − at least three versions for Unix, and at least two free ones for Windows. Version 1 of SSH is available for free for non−commercial use,*but is under a more restrictive license than the Gnu Public Licence.* It is maintained by SSH Communications Security Limited − although they intend to drop support for SSH 1 soon. Version 2 is being maintained and developed by the same people, including Tatu Yl˜ nen who originally wrote SSH − like SSH, it's available free for non−commercial and educational use, but the license is still not GPL.

There is also the OpenSSH project currently under way. It was developed by the OpenBSD people, under the OpenBSD licence. OpenSSH has port forwarding with the same command line as SSH 1, although X forwarding is disabled at installation for security reasons with the RPM packages of OpenSSH that we worked with. Most versions also rely on OpenSSL, so you should have that installed before you try to install OpenSSH.

In the Windows world, you can use TTSSH or PuTTY to connect to an SSH server. TTSSH supports port forwarding both from the GUI and the command line. PuTTY doesn't seem to support forwarding at all yet, but it's still beta. There are also pay versions of SSH for Windows available (primarily from F–Secure, who are directly associated with SSH Ltd.), but we won't be discussing these. We will be addressing SSH v1 on Unix, and to some extent connecting to Unix from Windows clients.

SSH is available from ftp.ssh.com:/pub/ssh as a tarball. *RedHat and other users of rpm packages might be able to get ssh and openssh from rpmfind.net, but major distribution vendors don't seem to be contributing, so make sure you trust the source of the package.* In the case of cryptography packages like this one, they used to link to www.replay.com. Replay is now Zedz.net, and rpmfind no longer links to them. To the best of our knowledge, Zedz.net's service is still sound: you can find packages at ftp://ftp.zedz.net/pub/crypto/redhat/i386/ . Binary packages are available for other distributions as well. If you're really serious about your cryptography, you'll get the sources, check the PGP signature, check the source for backdoors, and compile it yourself. However, that's a fairly arduous task, and not what we're here to discuss.

3. Basic Use of SSH

The most basic use of SSH is as a replacement for telnet and rsh. At a command prompt, just type "ssh hostname.com" instead of "telnet hostname.com". This will only work if hostname.com has the SSH server software installed. We encourage you, if you're a system administrator, to turn off your telnet servers, and switch completely over to SSH.

Login can be set up in two ways, either using a PGP key, or plain password. Either way, SSH never passes anything in the clear: two way handshaking and exchange of session crypto keys takes place before any passwords or passphrases are sent. After you send your password, the session behaves more or less like telnet, but it's encrypted at all times. SSH behaves in the same manner as rsh, in that it assumes your login name on the remote server is the same as on the local one. To override this behaviour, use something like this:

```
root@localhost$ ssh remotehost -l
giles
```

4. Port Forwarding

The concept of port forwarding is relatively simple. Unfortunately, the command line that accompanies it tends to be a little cryptic, and there are some twists and turns to work through. The idea is to allow any connections made to a port on the local machine to be sent through the encrypted SSH connection to the remote machine, so connections that would normally be clear text (such as FTP, SMTP, or NNTP, just to name a few) can benefit from SSH security.

5. Forwarding X

One of the great beauties of SSH is that X Forwarding can be (and usually is) enabled during compilation, so that after you have connected to another server with:

```
giles@tesla$ ssh remotehost

giles@remotehost’s password:

No mail.

giles@remotehost$
```

You can start X clients simply by typing them as you would at the console of the remote machine, and they will appear on your local machine:

```
giles@remotehost$ xclock &

[1] 26416

giles@remotehost$ xterm &

[2] 26422

giles@remotehost$
```

This is because SSH has already set up your local machine as the machine to display programs on:

```
giles@remotehost$ set | grep DISPLAY

DISPLAY=tesla.gcsu.edu:10.0

giles@remotehost$
```

If you want to start using your local machine as the work environment for your remote host, one of the cooler things you can do is start an application manager like the Gnome project’s "panel", or perhaps something like "tkdesk" or "xfm" if that’s more your style. If you’re not familiar with doing something like this, a word of warning: you have to be cautious about remembering where any given app has been launched from, either locally or remotely. If you’re reconfiguring the remote host, and you type "rm –rf /usr/local/bin/*" on the wrong machine, it will be very painful ... One solution for that is to colorize your prompts differently on every machine you work on, with the host name prominently displayed.

Often X forwarding is turned off on servers that have a lot of users. My ISP in Toronto, for example, has it turned off, presumably because their business is dial–up connectivity and web hosting, so they don't want the machine bogged down by fifty users all running Netscape on their processors. I tend to turn it on on my servers, but they usually only have twenty or thirty users, and most of those people are only using FTP.

6. Forwarding News

Now let's move on to the forwarding that you can do that isn't configured for you. Perhaps the easiest way to explain what this is about is to give an example that happened to me.

I live in Georgia, but I'm originally from Toronto. I'm still interested in what happens in Toronto, so I like to look at the Toronto news groups occasionally. So I set up port forwarding for NNTP, Network News Protocol, to allow me to read the news groups at my ISP in Toronto rather than on my local ISP, which doesn't carry Toronto groups. NNTP is carried on port 119, and you use some kind of news reader to look at it. I usually use trn – a bit arcane, but quite powerful.

The servers we're dealing with in this case are "shell.interlog.com" and "news.psi.ca". The former is my ISP's shell server, and the latter is their news server. I can't log in to the news server directly, it has no shell access. I can read news by logging in on shell.interlog.com and running trn, but I'd rather do that on my home machine. And since I can't make news requests from my home machine (it's not in the IP range news.psi.ca accepts connections from), I had to figure out how to make it work another way. This looks a bit daunting, bear with me.

```
giles@tesla$ su -c "ssh -C -L
    119:news.psi.ca:119
    shell.interlog.com -l giles"
```

I'm going to look at this starting at the back end. Since I'm using su to become root (I'll explain why in a second), ssh thinks I'm root on the remote machine. I don't have root on my ISP, so I have to tell it otherwise. Since I can't connect directly to news, I connect to shell. It isn't the machine that I'm getting news from, and that's one of the interesting twists here. Since "max3–42.dial.accucomm.net" (my assigned dial–up IP address) isn't authorized to get news from news.psi.ca, we connect to a machine that is authorized. I connect and port–forward through shell.interlog.com, which I have access to, and which is authorized to get news from news.psi.ca. So the "−L 119:news.psi.ca:119" is telling ssh to take port 119 on the local machine and send all information from there to port 119 on news.psi.ca. shell.interlog.com acts as a relay. The "−C" flag is important for me at home: I have a 56k dial−up connection, and "−C" means "use compression." This isn't useful if you have ethernet because the overhead of the compression slows you down more than the compression speeds you up, but on a phone line, it's great. Note that the connection isn't encrypted between shell and news, but that's not as important as having it encrypted on the Internet at large.

Finally, why did I su to root? Port 119 is a privileged port. On the remote machine, I'm only feeding data to it, so I don't need root, but on the local host, I'm taking it over entirely, so I have to be root. All ports below 1024 are privileged, so you have to have root to forward them. Often, if you don't have root, you can use another port above 1024, and tell the application you're using to look at the other port you chose − when you use "−L" with SSH, the two port numbers DO NOT have to be the same.

Now I'm ready to read news on my machine, as if I was directly connected to my ISP in Toronto. You need to set one environment variable on the local machine, and you're ready to go:

```
giles@tesla$ NNTPSERVER=localhost ;
    export NNTPSERVER
giles@tesla$ trn
```

That's it: you should be up and running.

7. Forwarding FTP

I recently put my web pages on another ISP in Toronto. Since I live in Georgia, my local ISP is quite a few hops away from my ISP in Toronto, and there are a lot of computers I don't trust between me and my new web host. So I use ssh to connect to them, and I would use scp (a secure replacement for rcp that's usually bundled with ssh) to copy my files to and from their service. Unfortunately, for reasons unexplained, they don't have scp (secure copy, a part of the SSH package) installed. So I decided to use port forwarding to use FTP to connect to them. In this case, it was a bit easier to do it as a user rather than as root:

```
giles@tesla$ ssh −C shell.eol.ca −L
    2121:ftp.eol.ca:21
giles@shell.eol.ca's password:
    Warning: Remote host denied X11
    forwarding, perhaps xauth program
    could not be run on the server
    side.
giles@eol$
```

Note the warning: you will see something like that when X11 forwarding is turned off. Now, in another window, I do something like this:

```
giles@tesla$ ftp localhost 2121
Connected to localhost.
```

```
220 babbage.echo-on.net FTP server
(BSDI Version 7.00LS) ready.
Name (localhost:giles): giles
331 Password required for giles.
Password:
230 User giles logged in, access
    restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

As soon as you issue that ftp command you’ll see a message in your other window that’s connected to the remote machine:

```
giles@eol$ fwd connect from 127.0.0.1
    to local port sshdfwd-2121
```

At this point, it looks like everything is cool and you’re in business. But ...

```
ftp> ls
200 PORT command successful.
425 Can’t build data connection:
    Connection refused.
ftp>
```

The problem is that FTP spawns connections on other ports by default to take care of the work. Since the other ports aren’t forwarded, it tends to get confused. This is the biggest "gotcha" I’ve found so far in SSH port forwarding, it took quite a while to figure this out.

So I issue the following command:

```
ftp> passive
Passive mode on.
ftp> ls
227 Entering Passive Mode
(205,189,151,4,203,122)
150 Opening ASCII mode data
connection for ’/bin/ls’.
total 134
...
```

If you’re using a GUI client, you may need to do some digging to figure out how to convince it to switch to passive mode. My preferred client these days is lftp, and I found out by digging through the man page that what I needed to do was this:

```
giles@tesla$ lftp -p 2121 -u giles
localhost
Password:
LFTP giles@tesla:~
> set ftp:passive-mode on
```

It would also be relatively easy to put a setting in your ~/.lftp/rc file so that all connections to local host were in passive mode by default.

Another way to establish the same connection would be to use the following command line:

```
ssh -f -C -L 2121:ftp.eol.ca:21
    shell.eol.ca "sleep 30"
```

There are only two differences here: the "−f" flag and the command "sleep 30" added on the end. "−f" requests ssh go to background after authentication is done and forwardings have been established. This will get you your terminal back, and "sleep 30" allows you enough time to set up a connection with ftp to keep the forwarding alive. It will die when you disconnect.

8. Port Numbers

You may be wondering how I know what port numbers to use. This is actually surprisingly easy to find out: "less /etc/services" *on a UNIX machine* will tell you a great deal about what ports are used for what purposes. I'll outline a couple more examples, but when we're done, if I haven't covered what you want to do, you should have the tools to figure it out yourself.

9. Forwarding Mail under Windows

Jacob and I work at a mid–sized university in middle Georgia, and all of the university dorms are wired. The university also has both a Computer Science program and a Business Information Systems program, so we have a lot of potential crackers live on our network before you even consider the fact that we have a T1 to the rest of the internet without a firewall ... So when I collect my mail using Eudora running on Windows, I'm not enthusiastic about having my password sent in clear text three or four times a day, or however often I press the "Check Mail" button. So I used a free Windows SSH client to secure my connection to our mail server – fortunately, we're allowed telnet and ssh access. The mail server administrator has set it up so that as soon as you log in, Pine starts up, and there's no way out to the shell (without getting very creative), but this doesn't matter: forwarding is taken care of on your local machine, so long as an SSH connection is established.

The program that I use is TeraTerm Pro, a very nice terminal emulator for Windows. It's available at http://hp.vector.co.jp/authors/VA002416/teraterm.html . You will also need the SSH extensions, available at http://www.zip.com.au/~roca/ttssh.html , which turn TeraTerm into an SSH client. Both of these are available for free. You can set port forwarding up in the "Setup" –> "SSH Forwarding" menu item, or you can do it from the command line in batch mode:

```
"C:\Program Files\TTERMPRO\ttssh"
    mail.gcsu.edu:22 /ssh
    /ssh-L110:mail.gcsu.edu:110
    /ssh-L25:mail.gcsu.edu:25
```

I then have to give a password login to the mail server. Since it's Windows, I don't have to give a root password to forward privileged ports.

The first /ssh switch tells ttssh to run in SSH mode. The next two are each just like the –L switches to ssh under Unix. I'm forwarding SMTP (port 25) and POP3 (port 110), so both sending and retrieving mail is encrypted. Sending mail this way isn't of much benefit if the mail is going out into the world, because it goes clear–text as soon as it's past the mail server, but the majority of my mail is going to recipients on the same mail server.

10. Pitfalls

Once SSH Forwarding is established on a Unix box, all users of that machine can – and in fact, have to – use the forwarding. If someone is trying to FTP to localhost (perhaps to access files as another user) while you have forwarding set up on that port, they'd get a big surprise, finding themselves connected to another server. This would be a pretty good argument for using a high number non–standard port if you're working on a multi–user system. However, it would occasionally be advantageous to allow other users on other machines to use a forward that you have set up: by default, SSH doesn't allow this, but you can switch it on if you like. The option is "–g". I would suggest using this with extreme caution.

11. Conclusion

SSH Port Forwarding is primarily a single–user pursuit, and any access by multiple users should be approached with considerable caution. Initial setup can also be a bit tricky, and we recommend that you create scripts to assist you in recreating the forwards you need. Within these limitations, SSH Port Forwarding is an extremely useful extension to the security provided by SSH, allowing you to encrypt many other applications that didn't originally present a security risk in an unencrypted form.

12. Resources

SSH Communications Security Limited is located at http://www.ssh.com/ . Source code for either version of SSH is available. You can also go directly to ftp://ftp.ssh.com/pub/ssh/ .

http://zedz.net/ has RPM and source packages for most of the Unix software discussed in this talk.

OpenSSH can be found at http://www.openssh.com/portable.html.

Putty can be obtained at http://www.chiark.greenend.org.uk/~sgtatham/putty/ : it is a beta Windows–based SSH client. At the time of writing, it doesn't support port forwarding, but it does include a Windows version of scp called pscp.

TTSSH is an add–on for Teraterm. Teraterm is an excellent Windows software terminal emulator, available at http://hp.vector.co.jp/authors/VA002416/teraterm.html . TTSSH is available at http://www.zip.com.au/~roca/ttssh.html . Commonly used ports (which you can check for yourself in /etc/services on most Unix machines) are given below.

```
ftp-data      20/tcp
ftp           21/tcp
ssh           22/tcp                      # SSH Remote Login
telnet        23/tcp
smtp          25/tcp     mail
www           80/tcp     http             # WorldWideWeb HTTP
pop-2         109/tcp    postoffice       # POP version 2
pop-3         110/tcp                     # POP version 3
nntp          119/tcp    readnews untp    # USENET News Transfer
imap2         143/tcp    imap             # Interim Mail Access
snmp          161/udp                     # Simple Net Mgmt Proto
irc           194/tcp                     # Internet Relay Chat
imap3         220/tcp                     # Interactive Mail Access
exec          512/tcp
biff          512/udp    comsat
login         513/tcp
who           513/udp    whod
shell         514/tcp    cmd              # no passwords used
syslog        514/udp
printer       515/tcp    spooler          # line printer spooler
talk          517/udp
ntalk         518/udp
uucp          540/tcp    uucpd            # uucp daemon
rsync         873/tcp                     # rsync
mysql         3306/tcp                    # MySQL
ircd          6667/tcp                    # Internet Relay Chat
```