1 SSH Port Forwarding and Tunnelling

SSH can be used for far more than simply a secure version of telnet / rsh. It is possible to securely forward TCP over it, to talk to CVS servers over it, syncronise data, and to run X11 programs over it.

1.1 SSH Proxy Command

It is possible to specify a command to use to connect to a server. This is useful for several reasons. You can "proxy" a SSH connection through another host to reach hosts you can't reach via normal methods. Alternatively, you can use a proxy server that supports CONNECT to tunnel ssh over it.

To proxy a ssh session over another ssh, there are two ways:

```
$ ssh -t host1 ssh -t host2
```

or put the following in your ssh config file (usually ~/.ssh/config, unless otherwise specified)

```
host host1 host2
        ProxyCommand ssh -q -a -x proxyhost nc %h 22
```

The second way is perhaps more useful, as you can also use scp with it.

To restrict where people can ssh, or if you don't want to give accounts out on the proxy box, you can emulate this using authorized_keys. Create a user on the proxy box, and put the users ssh keys in the authorized_keys file.

```
command="/usr/bin/ssh username@server" 1024 37 105798033118795863
580216860523001039809417647976823790615912077041585657005763456l5
859467315925629305341792897405316795971434191984242962247113922 90
022888329040775188174540488584967399729921726036019415856941697 89
398831603045743787860440848601735247314164144407641653524804350 96
497659551517839225252941542428 9 bmarshal@gosling
```

1.2 SSH Port Forwarding

Forwarding TCP over ssh is possible using port forwarding. This can be done in a couple of ways - either via command line, or by settings in a config file.

Command line:

```
$ ssh -L 8000:host1:80 host2
```

Config extract:

```
host host2
     LocalForward 8000 host1:80
```

This command forwards local port 8000 to port 80 on host1.

1.3 CVS over SSH

To access CVS remotely, the traditional method is using CVS's pserver. This is insecure, as it is a plain text password. It is possible to tunnel CVS over ssh, both if the users have an account on the box, and if they don't.

If the user doesn't have an account on the box, it can be done using the authorized_keys file.

```
command="/usr/bin/cvs server" 1024 37 105798033
118795863580216860523001039809417647976823790 61
591207704158565700576345615859467315925629305 34
179289740531679597143419198424296224711392290 02
288832904077518817454048858496739972992172603 60
194158569416978939883160304574378786044084860 17
352473141641444076416535248043509649765955151 78
392252529415424289 bmarshal@gosling
```

Regardless, to use CVS over ssh, use something like the following.

```
$ export CVS_RSH=ssh
$ export CVSROOT=:ext:username@host:/path/to/cvsroot
$ cvs co module
```

1.4 Rsync over SSH

Rsync is a program that is used to transfer files between hosts—much like scp or rcp—but uses an efficient protocol that means only the differences are transferred. By default, it uses rsh as a transport layer but can be configured to use ssh rather easily.

This is done either by setting the RSYNC_RSH environment variable to point to your ssh binary, or by using the -e command line option. Further details about rsync are available from the rsync(1) manpage.

2 Conclusion

As you have seen, ssh is much more than a secure version of telnet. It is able to encrypt just about any kind of TCP connection with very little setup on either side, other than installing ssh.

For more information, see the ssh(1), sshd(8), cvs(1), and rsync(1) manpages, as well as the OpenSSH, CVS and Rsync websites.