The Secure Shell

by John S. Quarterman

In last month's column ("Cryptography and the Internet," December 1996, Page 25), we mentioned ssh, the secure shell, in passing. Let's look at ssh in more detail. Those of you who are waiting for the politics of cryptography and the Internet, don't worry, that's next month.

If you want to transfer a file securely across an insecure network, and you can arrange to set up a software package on each end, scp is a convenient solution. This is the file transfer program that comes with ssh. You can use it with exactly the same syntax as rcp to transfer files between two systems.

One difference is that, unlike rcp, scp will prompt for a password or user name if it needs one for authentication. A little inconvenience is the cost of more security; that is the trade-off with any security package.

Uses of ssh

Perhaps more common than scp is the use of ssh itself. With ssh you can log in on a remote system, just as you can with rlogin or Telnet. The difference is that with ssh you get a high degree of security regardless of the security, or lack thereof, of the intervening transit path. This is quite useful for connecting to your home system from a conference or hotel.

I use ssh all the time to connect one of the PCs at home to the office machines. That may seem overly paranoid because I don't go over any public parts of the Internet to get there, but I think using ssh is a good habit to get into.

I can attest that the performance penalty is so minor that I don't notice it even when I'm using a slow modem connection. The initial connection

John S. Quarterman,of Texas Internet Consulting, is a consultant in networking and open systems, specializing in TCP/IP networking, UNIX systems and standards. He is also a principal in Matrix Information & Directory Services Inc., which publishes Matrix News *and* Matrix Maps Quarterly. *John has co-authored two recent books with Smoot Carl-Mitchell on networking:* Practical Internetworking with TCP/IP and UNIX *and* The Internet Connection: System Connectivity and Configuration, *both published by Addison-Wesley Publishing Co. John can be reached by email at* `jsq@tic.com`. *He can also be reached by voice at (512) 451-6176 or by fax at (512) 452-0127.*

setup is quite slow, but the actual connection is the same speed as a TCP connection over the same communications medium. Basically, CPUs are so much faster than communications links these days that encryption isn't much of a burden.

As the Internet expands, more unusual reasons for using ssh have been reported. In some countries, there is a tradition of government surveillance of all communications media, and such governments don't hesitate to apply the same philosophy to the Internet. We can only hope that tradition is alien to the United States. But even here, one can never be sure who might be listening: A playful fellow employee at the office, a cracker who broke into the local Internet service provider, a student with more time on his hands than brains, or a neighborhood child who's discovered the test jack on your telephone box. We certainly have all of those, even though our government agencies are supposedly more domesticated than in some other countries.

Is the added security worth the added complexity and slight inconvenience? You'll have to decide that.

Why ssh in particular? In short, because it handles several security problems that are common to other methods. Basically, nothing is sent in the clear; encryption is started before authentication, so passwords or passphrases are hidden. Passphrases can be different from local login passwords, and can be different for each pair of client and server machines, at the user's discretion.

Unlike rsh and rcp, authentication is not based solely upon DNS lookups. Matching encrypted keys is also required. So even if someone takes over a system and runs a fake domain server, they can't pretend to be you. Both the host and the user are authenticated using encrypted keys.

It does no one any good to tap into your ssh connection at any time by tapping your telephone or your Ethernet, for example, because the entire connection is encrypted.

The whole ssh package is rather cleverly packaged. In particular, the server, sshd, arranges to pass X Window System connections through the encrypted data channel. It does this partly by setting the DISPLAY environment variable to point to a socket it serves. The client end cooperates in handing the data off to X again.

Any TCP port can be handled this way, and thus any TCP application. In practice, I haven't found a lot of use for this flexibility, but it is good to know it is there if needed.

Where to Get ssh

ssh comes from Finland. There are numerous mirrors and ancillary repositories for the software itself. Because this is security software, you don't want to get it from just anywhere. Consult the ssh Web page, `http://www.cs.hut.fi/ssh/`. The U.S. locations recommended by the Web page as of this writing were `ftp://ftp.gw.com/pub/unix/ssh/`, `ftp://ftp.net.ohio-state.edu/pub/security/ssh/` and `ftp://ftp.neosoft.com/systems/unix/security/ssh/`. Tatu Ylonen is the primary instigator of ssh.

Finland is not only one of the more densely networked places on the planet, but is also one of the most productive. Having produced any one of Linux, `anon.penet.fi` or ssh would be adequate for international fame for a country of 5 million people, but Finland produced all three. What is it about that place?

Installation

Installation configuration is done using the Free Software Foundation's configure script. (Larry Wall should get a medal for inventing configure many years ago.) You can manually specify several options, which determine such things as whether to use IDEA or RSA. These options may be useful under certain legal conditions. Read the INSTALL file for details.

Another option permits you to compile ssh so that it can be used under the names rsh, rlogin and rcp, thus completely replacing those commands. If called by one of those names, ssh will use ssh encryption where the remote system supports it, but will fall back to using rsh or the like where necessary. This will require a bit more setup because you'll need to save the old rsh, rlogin and rcp in a directory where ssh can find them. You'll also need to save host keys for all your hosts in `/etc/ssh_known_hosts` if you want `/etc/ hosts.equiv` and `.rhosts` to have effect when using the ssh protocol among your hosts.

Next, type

```
make
make install
```

However, you will probably need to be root to do that last step, so it may really be:

```
sudo make install
```

Even without root access, you can run a copy of sshd to listen on a user-level port, and then point your ssh client to that port. It is better to use a root copy of sshd that is installed to restart whenever the system reboots.

If you install normally with root access, the sshd server program ends up in some place like `/usr/local/sbin/sshd`. It needs to be run standalone, not through `inetd`, because it can take quite a noticeable amount of time (many seconds) to compute the RSA key for the connection. This is the most noticeable performance penalty for using ssh. You must modify `/etc/rc.local` (BSD-style systems) or create an `/etc/init.d/sshd` (System V-style systems) startup file such as this one:

```
#!/bin/sh
#
# start sshd daemon
sshd=/usr/local/sbin/sshd
case "$1" in
    if [ -f $sshd ] ; then
```

```
        echo "starting sshd"
        $sshd
    fi
;;
    kill `cat /etc/sshd_pid`
;;
esac
```

The Makefile sets up most of the parameter files you will need.

The basic server configuration file is /etc/sshd_config, and the basic client configuration file is /etc/ssh_config. These files set such parameters as the port to use, which is normally port 22, the ssh port that is registered with IANA.

In /etc/sshd_config you can also set hosts to allow or deny, using literal names or wild card syntax similar to that used by tcpd. You may want to do this. For example,

```
AllowHosts *.yourdomain.com
DenyHosts *.dialups.isp.net
```

These lines can be used to severely limit which remote hosts can connect to your system. Most of the other parameters in /etc/sshd_config are fine for most systems without modification.

As part of the installation, make runs ssh-keygen to create the file /etc/ssh_host_key, which then contains a private key for the host.

The server looks at /etc/hosts.equiv for clues as to what hosts to permit but, unlike rshd, sshd will not permit a connection solely on the basis of the contents of /etc/hosts.equiv. If you want /etc/hosts.equiv or .rhosts authentication to work, you need to set up the file /etc/ssh_known_hosts to store the public keys of the remote hosts for which you want /etc/hosts.equiv or .rhosts to work.

The ssh distribution comes with a Perl script, make-ssh-known-hosts.pl, that can produce appropriate contents for /etc/ssh_known_hosts. Normally, you call it with a domain name, for example,

```
perl5 make-ssh-known-hosts.pl yourdomain.com > /etc/ssh_known_hosts
```

The script uses DNS to find all the hosts in your domain, and queries each one individually with ssh to get its public key.

Client Host Configuration

Setting up ssh is pretty boring, actually, because the combination of configure and make install does almost everything for you. The /etc/ssh_config file for the client is even more boring, because normally it has nothing at all set–defaults that are compiled into the binary are used instead.

You normally want ssh to be installed setuid root, but you can use it without that. However, you won't be able to take advantage of personal authentication files.

You can tailor your use of ssh with your own authentication files. On the server side, .rhosts works more or less as with the rsh commands. You can also have a .shosts file with the same syntax. The advantage of .shosts is that it permits ssh to connect without permitting rsh to connect.

Basically, to use ssh you need an authentication key. You can generate this with ssh-keygen. The private key normally goes in $HOME/.ssh/identity and the public key goes in $HOME/.ssh/identity.pub. The ssh-keygen program asks you where to put these keys, but the default location is good enough.

Also, ssh-keygen asks you for a passphrase. This can be a song lyric, a sentence, arbitrary characters or whatever you find easy to remember, but that is both hard to guess and long enough to be hard to break by iteration. You will normally be prompted to give this passphrase when you connect with ssh or scp.

You can also use ssh-keygen to generate host keys, but be careful *not* to supply a pass phrase.

Once you've got your personal public key, you need to put a copy of it in $HOME/.ssh/authorized_keys on each system you intend to connect to with ssh. The authorized_keys file can contain several keys, each for a different client machine.

Remote host public keys are recorded by sshd in $HOME/.ssh/known_hosts. The program will add a host to that file once you've successfully connected from that remote host with sshd. If /etc/ssh_known_hosts exists, sshd will check it first. The client, ssh, also looks at these two files to verify the identity of the server machine.

Finally, you can have your own $HOME/.ssh/config file that can override parameters set in /etc/ssh_config. For example, you could use $HOME/.ssh/config to tell ssh not to fall back to rsh if the remote system does not support the ssh protocol.

Although ssh is clearly of UNIX origins in its network orientation, sophistication and subtlety, and it runs under almost every known modern version of UNIX, it also runs under Windows and OS/2. A Macintosh port is expected soon. This means you can have ssh on a random PC, without even installing a UNIX system on it. Theoretically, I suppose you could carry a floppy with ssh and use it on an arbitrary PC.

For More Information

There is an ssh mailing list. To get on it, send a message like this:

```
To: majordomo@clinet.fi
Subject: whatever
subscribe ssh
```

The contents of the Subject: header are irrelevant. The command to subscribe goes in the body of the message. If you prefer to get announcements without all the chatter, do this instead:

```
To: majordomo@clinet.fi
Subject: whatever
subscribe ssh-announce
```

▲