NSRC

Network Startup Resource Center

Security with SSH

Topics

- Where to get SSH (Secure SHell)
- How to enable and configure SSH
- Where to get SSH clients for Windows
- Authentication of the server to the client (host keys)
- Issues to do with changing of the host key
- Password authentication of the client to the server
- Cryptographic authentication of the client to the server (rsa/dsa keys)
- hostkey exchange, scp, and sftp labs

Main Security Concerns

SSH applies directly to dealing with these two areas of security:

- Confidentiality
  - Keeping our data safe from prying eyes
- Authentication and Authorization
  - Is this person who they claim to be?

Where to Get the SSH server

First see if SSH is installed on your system and what version. Easiest way is:

```
$ ssh -V
```

If you want or need an updated version of OpenSSH (current version is 5.9) you can simply type:

```
$ apt-get update
$ apt-get install openssh-server
```

We recommend using OpenSSH for Ubuntu. Default version installed in FreeBSD 12.04 LTS is OpenSSH Portable version 5.9p1

Enable and Configure OpenSSH

On our machines this is already done, but if you just installed it you may want to see how it’s configured.

Take a look at /etc/ssh/ssh_config and /etc/sshd_config.

In sshd_config you might be interested in:

```
#   PermitRootLogin yes/no (you generally want “no”)
```

We'll be allowing root login, but only with keys in our exercises.

There are many options in ssh_config and sshd_config. You should read through these files to verify they meet your expecations.

Where to Get SSH Clients for Windows

There are several free, shareware, and commercial ssh clients for Windows. See http://www.openssh.org/windows.html for a list.

A few that support protocol version 2 include:

- Putty: http://www.chiark.greenend.org.uk/~sgtatham/putty/
- Kirby from http://www.downloadbestsoft.net/programs/kitty_portable.exe
- OpenSSH for Windows (using Cygwin): http://www.cygwin.com/ http://sshwindows.sourceforge.net/
- Secure Shell from ssh.com (free for personal use): http://www.ssh.com/products/ssh/download.cfm
- And WRQ at http://www.wrq.com/products/reflection/ssh/ is a nice product if you are willing to pay.

SSH Connection Methods

Several things can happen when using SSH to connect from your machine (client) to another machine (server):

- Server's public host key is passed back to the client and verified against known_hosts
- Password prompt is used if public key is accepted, or already on client, or
- RSA/DSA key exchange takes place and you must enter in your private key passphrase to authenticate (assuming you have one).

SSH Quick Tips

- You have a choice of authentication keys - RSA is the default (dsa is fine as well).
- The files you care about are:

```
/etc/ssh/ssh_config
/etc/ssh/sshd_config
~/.ssh/id_dsa and id_dsa.pub
~/.ssh/id_rsa and id_rsa.pub
~/.ssh/known_hosts
~/.ssh/authorized_keys
```

- And, note the rsa/dsa host-wide key files in /etc/ssh
- Be sure that you do “`man ssh`” and “`man sshd`” and read the entire descriptions for both the ssh client and ssh server (sshd).

SSH Authentication

Private key can be protected by a passphrase

So you have to give it each time you log in or use "ssh-agent" which holds a copy of your passphrase in RAM

No need to change passwords across dozens of machines

Disable passwords entirely!

```
vi /etc/ssh/ssh_config
#   PasswordAuthentication yes
```

Man in the Middle Attacks

The first time you connect to a remote host, remember its public key

Stored in ~/.ssh/known_hosts

The next time you connect, if the remote key is different, then maybe an attacker is intercepting the connection!

Or maybe the remote host has just got a new key, e.g. after a reinstall. But it's up to you to resolve the problem

You will be warned if the key changes.

Exchanging Host Keys

- First time connecting with ssh:

```
ssh username@pc1.cctld.pacnog2.dnsdojo.net

The authenticity of host 'pc1.cctld.pacnog2.dnsdojo.net (202.4.34.65)' can't
   be established.

DSA key fingerprint is 91:ba:bf:e4:36:cd:e3:9e:8e:92:26:e4:57:c4:cb:da.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'pc1.cctld.pacnog2.dnsdojo.net, 202.4.34.1' (DSA)
   to the list of known hosts.

username@pc1.cctld.pacnog2.dnsdojo.net's password:
```

- At this point the client has in the file ~/.ssh/known_hosts the contents of pc1.cctld.pacnog2.dnsdojo.net's /etc/ssh/ssh_host_dsa_key.pub.
- Next connection:

```
[hallen@hallen-lt .ssh]$ ssh usrname@pc1.cctld.pacnog2.dnsdojo.net

username@pc1.cctld.pacnog2.dnsdojo.net's password:
```

- Now trusted - Not necessarily a good thing...

More SSH References

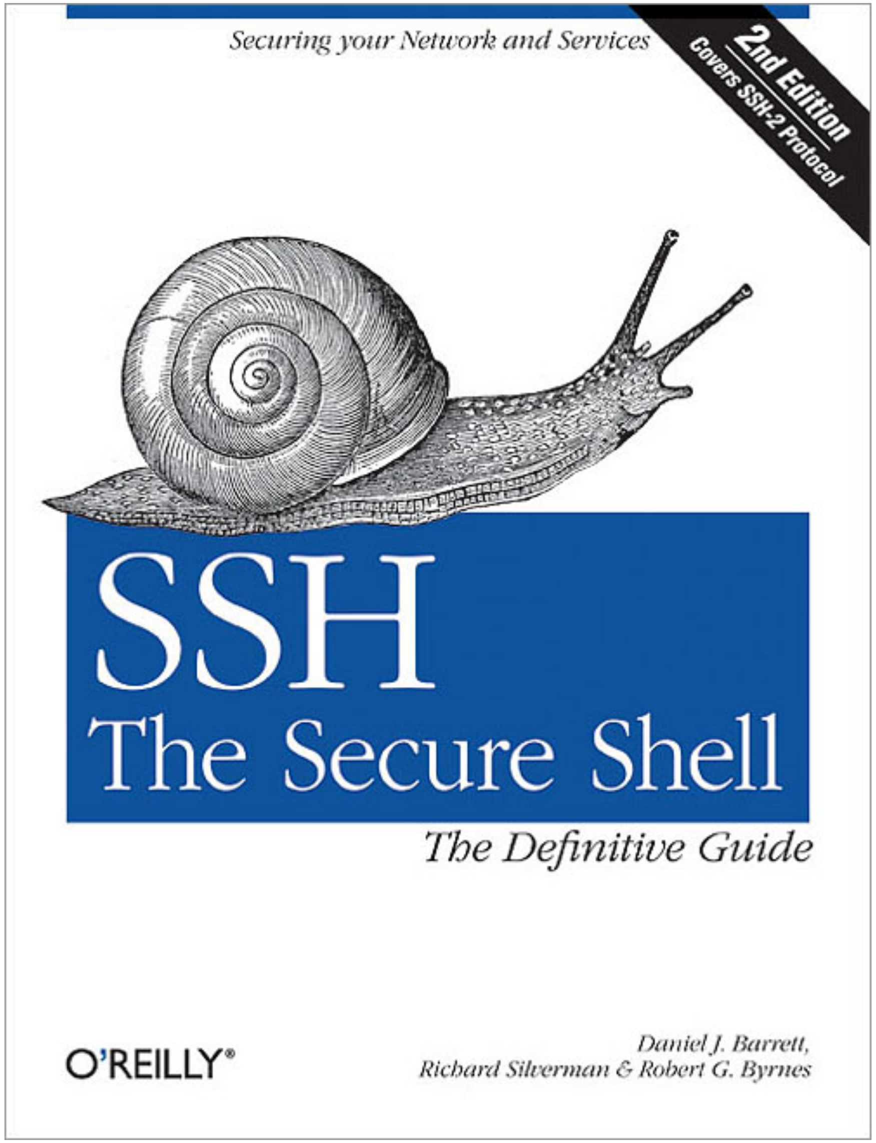
For a comparison of SSH Version 1 and 2 see: http://www.snailbook.com/faq/ssh-1-vs-2.auto.html

An excellent book on SSH is:

SSH, The Secure Shell
The Definitive Guide,
Second Edition.

By
Daniel J. Barrett,
Richard Silverman, &
Robert G. Byrnes

May 2005
ISBN: 0-596-00895-3



Exchanging Host Keys Cont.

Command	Key Type Generated	Public File
`ssh-keygen -t rsa`	`RSA (SSH protocol 2)`	`id_rsa.pub`
`ssh-keygen -t dsa`	`DSA (SSH protocol 2)`	`id_dsa.pub`

- Default key size is 1024 bits
- Public files are ordinary text files
- Private files are encrypted if you use a passphrase (still text)

Corresponding file on the host for host key exchange is “known_hosts”.

Exchanging Host Keys Cont.

How does SSH decide what files to compare?

Look in /etc/ssh/sshd_config. For OpenSSH version 3 the server defaults to protocol 2 .

By default OpenSSH version 2 client connects in this order:

- RSA version 2 key
- DSA version 2 key
- Password based authentication (even if RSA version 1 key is present)

Pay attention to the “HostKeyAlgorithms” setting in /etc ssh/ssh_config to help determine this order - or use ssh command line switches to override these settings.

SSH - “Magic Phrase”

Basic concept to understand how an SSH connection is made using RSA/DSA key combination:

- Client X contacts server Y via port 22.
- Y generates a random number and encrypts this using X's public key. X's public key must reside on Y. You can use scp to copy this over.
- Encrypted random number is sent back to X.
- X decrypts the random number using it's private key and sends it back to Y.
- If the decrypted number matches the original encrypted number, then a connection is made.
- The originally encrypted random number sent from Y to X is the “Magic Phrase”

Tunneling with SSH

We'll do this if there's time and interest...

:-)

==>

Exercises

Now I'll ask you to do the following

- Ensure you all have SSH client software installed on your machines.
- Ensure you all have VNC clients installed too on your machines
- Try reaching the DMZ machine in your group using SSH

Tunneling with SSH

You can use SSH to tunnel insecure services in a secure manner.

- SSH tunneling services includes authentication between known_hosts, password challenge, and public/private key exchanges.
- You can even indirectly tunnel via an intermediary machine.

Tunneling with SSH Cont.

The basic concept looks like this:

- Connect from one machine to another as username.
- Use ssh options to specify the port number on the remote machine that you wish to forward to the port on your local machine.
- Your ssh connection will “tunnel” data securely across ssh from the remote machine to your local machine.

There are several options to be aware of.

Tunneling with SSH Cont.

Tunneling by Example

Here is a sample tunnel command using SSH under FreeBSD:

```
ssh -C -f username@host.domain -L 1100:localhost:110 sleep 10000
```

What is happening here?

- The '`-C`' option specifies compress the data. Good if it works.
- '`-f`' means ssh goes to the background just before executing the specified command listed (in this case, “sleep 10000”).
- '`-L`' forwards the port on the left, or client (1100) to the one on the right (110) or remote side.

Tunneling with SSH Cont.

Tunneling by Example Cont.

So, what does this command do?

```
ssh -C -f username@host.domain -L 1100:localhost:110 sleep 10000
```

This “tunnels” your POP email from port 110 on the remote side through port 1100 on your local side.

The process backgrounds for 10000 second (detaches and runs).

This is done under the authority between yourself (client) and user@host.domain.

Diagram* of Tunneling both smtp and POP Services

Tunneling with SSH Cont.

Tunneling by Example Cont.

Why use something like ports “1100” and “2500”?

- Ports up to 1024 can only be bound to by the admin user.
- If you are admin you can forward 110 to 110, 25 to 25, and so on.
- Other popular tunneling tricks include tunnels for XWindows, IMAP, etc.
- On the client side you must set programs to use “localhost” – For example, for POP and smtp, your mail client must use “localhost” instead of host.domain (i.e. no more “mail.host.com”).
- If you are not admin, and your ports are changed, then your mail client must be able to set the smtp and POP ports as well.
- We may show or discuss this using a local email client now.

Tunneling with SSH Cont.

One More Tunneling Example

- You can use SSH to do “Indirect Port Forwarding”
- What to do if your organization's email sits behind a firewall?
- Connect via an intermediary box (gateway).
  Here's a real world example:

```
ssh -C -f hallen@gateway.turbolinux.com -L
2500:mail.us.tlan:25 -L 1100:mail.us.tlan:110 /bin/sleep
10000
```

Tunneling with SSH Conclusion

- Tunneling lets you securely access basic services such as POP and IMAP.
- You can securely tunnel ports using SSH.
- You can use /etc/services to verify you are not using a port that is already defined.
- Only admin can redfine ports below 1024.
- You can tunnel ports directly between two machines, and indirectly with a machine in the middle.