Security with SSH

Dean Pemberton
Network Startup Resource Center
dean@nsrc.org

Topics

- What is SSH
- Where to get SSH
- How to enable and configure SSH
- Where to get SSH clients for Windows
- Host keys: authentication of server to client
- Issues to do with changing of the host key
- Password authentication of client to server
- Cryptographic authentication client to server
- hostkey exchange, scp, and sftp labs

What is SSH?

From Wikipedia:

Secure Shell (SSH) is a cryptographic network protocol for secu data communication, remote command-line login, remote command execution, and other secure network services betwee two networked computers that connects, via a secure channel o an insecure network, a server and a client (running SSH server SSH client programs, respectively).

i.e., ssh gives you a secure command line interface on remote machines...

Topics

- Where 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?
  - *With keys alternative method to passwords*

Where to get SSH

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

```
$ ssh –V
```

- Commonly used SSH in Linux and FreeBSD is OpenSSH. You can find the home page here:

  http://www.openssh.org/

- You can install OpenSSH via packages on Linux and FreeBSD. Ubuntu 12.04.3 LTS currently installs vers 5.9p1 of OpenSSH.

Obtain SSH Client for Windows

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

Two free clients:

Putty:http://www.chiark.greenend.org.uk/~sgtatham/putty/

Secure Shell from ssh.com (free for personal use):
http://www.ssh.com/products/ssh/download.cfm

We will use Putty in this class and it is available on our local server.

Enable & Configure OpenSSH

FreeBSD

- `/usr/ports/security/openssh-portable/make install`
- You should make sure that /etc/rc.conf is set:

```
sshd_enable="YES"
```

Linux

- 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 allow root login, but only with keys in our exercises.

Many options in ssh_config & sshd_config. Read through these files to ve
they meet your expectations.

Some Useful SSH Resources

If you want a great SSH RSA/DSA key overview Daniel Robbins ex-CEO of gentoo.org has written a 3-part series hosted on the IBM Developer Works pages.

The three papers and URL's are:

OpenSSH Key Management, Part 1
http://www-106.ibm.com/developerworks/library/l-keyc.html
OpenSSH Key Management, Part 2
http://www-106.ibm.com/developerworks/library/l-keyc2/
OpenSSH Key Management, Part 3
http://www-106.ibm.com/developerworks/library/l-keyc3/

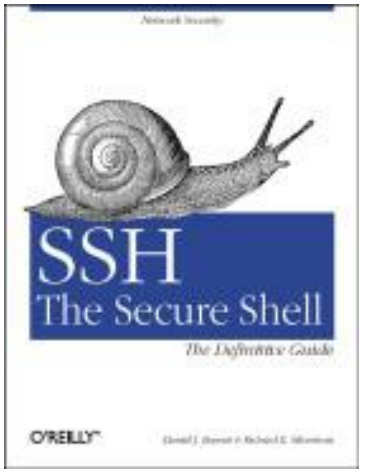
More SSH References

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



SSH Connection Methods

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

Server's public host key is passed back to the client ar
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

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 o machines
- Disable passwords entirely!
- /etc/ssh/ssh_config
- # PasswordAuthentication yes

Man in the Middle Attacks

The first time you connect to a remote host, its public key is stored in ~/.ssh/known_hosts

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

Or maybe the remote host has just got a new ke e.g. after a reinstall. But it's up to you to resol

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...

Exchanging Host Keys (continued)

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 text

Private files are encrypted if you use a passphrase

Corresponding file on host for host key exchange is "known_hosts".

Exchanging Host Keys (continued)

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 v2 client connects in 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

How an SSH connection is made using RSA/DSA key combo:

- 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

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

SSH tunneling services includes authentication
between known_hosts, password challenge, an
public/private key exchanges.

You can even indirectly tunnel via an
intermediary machine.

Tunneling with SSH: Concept

Connect from one machine to another as usernam

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 loc machine.

There are several options to be aware of.

Tunneling with SSH: By Example

Sample tunnel using SSH under FreeBSD:

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

What is happening here?

'-C' option specifies compress the data. Good if it works.

'-f' means ssh goes to the background just before executing t

specified command listed (in this case, "sleep 10000").

'-L' forwards the port on the left, or client (1100) to the one on

right (110) or remote side.

SSH Tunneling Example (continued)

So, what does this command do?

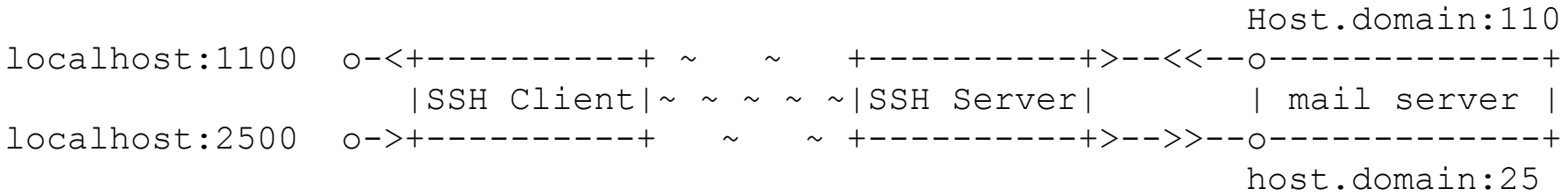
```
ssh -C -f username@host.domain -L 1100:localhost:110 sleep 10000
```

This "tunnels" your POP email from port 110 on the remote side through port 1
your local side.

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

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

Diagram of Tunneling both smtp and POP Services



SSH Tunneling Example (continued)

Why use something like ports "1100" and "2500"?

Ports up to 1024 can only be reset 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, e

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.domain").

If you are not admin, and your ports are changed, then your mail clien must be able to set the smtp and POP ports as well.

We may show or discuss this using a local email client now.

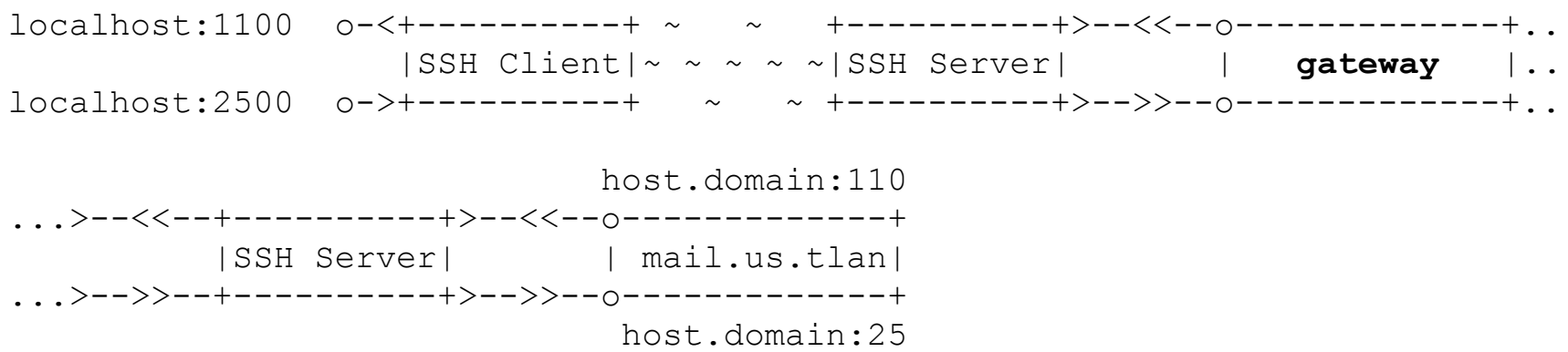
SSH Indirect Port Forwarding

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
```



SSH Tunneling Conclusion

Tunneling lets you securely access basic services suc
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 redefine ports below 1024

You can tunnel ports directly between two machines,
and indirectly with a machine in the middle.