Security with SSH

PacNOG10

Noumea, New Caledonia

John Kemp

Outline

Installing SSH

Enable/Configure SSH

Clients for Windows

Client/Server (Host) Authentication

Client/User (Key) Authentication

Cryptography

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 SSH

By default, most Unix systems will have at least the SSH client installed. In Ubuntu:

```
% ssh -V
% dpkg -l | grep ssh
% ps -ef | grep sshd
% sudo apt-get install openssh-server
% sudo apt-get install openssh-client
( same as saying )
% sudo apt-get install ssh
```

NOTE: **ssh** is the client, **sshd** is the daemon/server

Enable and Configure OpenSSH

You should make sure that ssh is enabled:

NOTE: this job is an “upstart/init” job so...

% sudo service ssh status
% ls -l /etc/init/ssh.conf
% ls -l /etc/init.d/ssh

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

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

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/

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

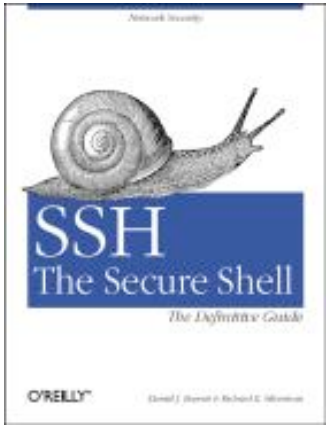
More SSH References

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



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

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!

- /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

Exchanging Host Keys

First time connecting with ssh:

ssh username@pc1.ws.nsrc.org
The authenticity of host 'pc1.ws.nsrc.org (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.ws.nsrc.org, 202.4.34.1' (DSA) to the list of known hosts.
username@pc1.ws.nsrc.org's password:

At this point the client has in the file ~/.ssh/known_hosts the contents of pc1.ws.nsrc.org's /etc/ssh/ssh_host_dsa_key.pub.

Next connection:

[hallen@hallen-lt .ssh]$ ssh usrname@pc1.ws.nsrc.org
username@pc1.ws.nsrc.org's password:

Now trusted - Not necessarily a good thing...

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 text
- 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”

Exercises

Now I'll ask you to do the following

Create public/private keys and copy them between neighbor machines

Copy your public key to /root/.ssh on neighbor's machine

Coordinate with your neighbor to update /etc/ssh/sshd_config

Consider the power of scp -r

Tunneling with SSH

The Topic You've Been Waiting For...

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 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.