Some subtleties of SSH (Secure Shell)

Alan Robert Clark

October 21, 2019

1 Introduction

Remote execution is the mainstay of any UNIX system. I realise that this is a novel idea to Bill's friends, but the simple ability to run an app on another machine must surely be a cornerstone of UNIX.

THE conrnerstone of UNIX *IS* the shell. Period. Pointy-Clicky stuff is very sexy, but is devoid of power: try putting it in a cron() job :-) :-)

Hence the Remote Shell: rsh(). The ability to start a shell on another machine. Plagued by security holes etc, due to ancient architecture. Enter ssh(). The Secure Shell: has occassional holes, but get filled rapidly.

A bit harder to understand though, and can be frustrating in terms of continually asking for passwords, when the encryption is supposed to be transparent!

2 Solution

I have solved the ssh password problem after a good three years of typing passwords for every ssh and scp I issue!!!!!!!!!

Forget the .shosts or hosts.equiv or etc etc etc—garden path.

sshd -d is a **wonderful tool**. Exactly lists the reasons for rejecting anything, and verbosely tells you what it is about to try.

Hence: (in /home/clark) (#= root, $ as clark) First generate your host keys, public and private. (2018 update ecdsa, not dsa)

```
#rm -rf ~/.ssh to clear the cobwebs. rm /etc/ssh/ssh_host*
ssh-keygen -t ecdsa
to /etc/ssh/ssh_host_ecdsa_key (NOT DEFAULT)
It provides the .pub's. NO PASSPHRASES (just press Enter)
Perms: 600 moduli, sshd_config, ssh_host*(non.pub)
       644 ssh_config, *.pub
```

Then, the user keys:

```
$ssh-keygen -t ecdsa as above (accept default files)
NO PASSPHRASES
vi .ssh/authorized_keys and :r the .ssh/id_ecdsa.pub
chmod 600 .ssh/authorized_keys
check 700 perms on .ssh
```

On the opposite machine, put this public key into the user's home `.ssh/authorized_keys` via the last password `scp()` you will do :-)

I was focussing on host based stuff, unnecessary. Protocol2 does a full public/private key tunneling, hence making a passphrase useless. I initially got Protocol1 working (rsa1), but got rsa and dsa working easily. DSA is preferred, so you can JUST do DSA.

3 years of passwords, sheesh. The docs are unclear though, giving too many options/decisions without rationale. A simple HOWTO (like this) is required!

Just thought I would let you know. sshd -d reveals the main reason for the documented methods' failure: permissions :-)

3 Getting remote X services...

This was baffling, since setting the `$DISPLAY` variable is tricky over a home fibre connection, through DUN etc.

All seamlessly handled via `ssh -X ytdp`. So simple.