Remote Access – Overview Sheet

Remote Access

When looking after a lot of machines, it's not always possible to be able to gain physical access to the keyboard and mouse plugged in (if there is one) or see the screen attached. This means that systems administrators will be required to access their servers and client desktops remotely to fix problems and run diagnostics.

Secure Shell (SSH)

SSH is the protocol and set of tools that allow your command line to log in to the command line of another machine. It transmits anything you type to the target computer and can also transmit other things, such as files, graphical interfaces and other data.

In order to secure the shell the host and client connections are encrypted with a number of layers. The first is the standard host verification key encryption and the second is the user authentication key encryption. Each layer is added security.

Host Verification

When you install an SSH server on a computer it will create a random key in /etc/ssh/ssh_host_*_key files, these not only allow for encryption but also varify that the computer your talking to is the one you expected. The SSH protocol will save the host key on first connection only when confirmed by the user.

If the SSH connection detects an incorrect verification key it will terminate the connection as this indicated a man in the middle attack. All keys are stored per user in ~/.ssh/known_hosts, if the host changes then you can manually remove it by editing this file.

Public/Private Encryption

A Public/Private key is a password protected key in two parts, the first is a public half which can be transmitted to anyone without fear of compromise. This half can only encrypt but not decrypt data. The private key can decrypt the message once it's been encrypted by a public source. This allows the user of these keys to receive secured messages from a variety of people while not having to share the private key with anyone.

As well as being able to type in your password for the target computer, you can also use a public private key. This will allow your computer to verify it's user to the target without the need for a password that gets transmitted to the client.

Instead you may be required to type in a password to unlock the local private key. This password is more secure because it never leaves your computer and is a very good idea to have.

Setting up the SSH Server

The default ssh server on ubuntu is openssh-server and is not installed by default. This package should be installed on all computers you want to access using apt-get. The service daemon script to make sure it's running is /etc/init.d/ssh, this should be running in order to successfully access that machine.

```
sudo apt-get install openssh-server
sudo /etc/init.d/ssh start
```

The client tools are installed by default, but for reference they are in the package openssh-client, there are manual pages for each of the available ssh tools.

Getting into another machine

To get into another machine running openssh server you simply use the `ssh [user@]host` command, this will attempt to connect to the target host on port 22 and using the user specified or the same one you are currently, it will exchange keys or challenge you with a password.

Once successful you will be able to run commands as if you were actually on that computer. You will notice that the host name on the command has changed and the way the computer behaves might also be different.

```
ssh student@192.168.50.14
```

Passwordless Access

To get into a machine without a password you will first have to create a public/private key for your ssh sessions. You can do this using the `ssh-keygen` command and specify the type of key to make. At the moment it's recommended you use dsa keys:

```
ssh-keygen -t dsa
```

This command will ask you a couple of questions, one will be a password with which to lock the private key. This is optional but recommended for securing the key, but this password will need to be typed in each time you want to access target computers.

The keys get generated and placed in ~/.ssh/ as two files id_dsa and id_dsa.pub, the first is the private key and should only exist on your own secure machine and the second is the public file which contains a single line, this should be copied to the ~/.ssh/authorized_keys file for each of the hosts you want to access, under the user you want to be able to use the key.

You can use a command called scp (ssh copy) to copy the file over, or you can simply copy the public key line onto the clipboard. You will need to gain access to the target host using standard password authentication in order to place your new key in the authorized_keys file. Create it if it doesn't already exist.

```
Scp ~/.ssh/dsa_key.pub student@192.168.50.14:~/.ssh/authorized_keys
```

Saving the Key Session

Because you now have to type in a password to unlock the private key, this is not “passwordless” in order to remove this barrier we have to save the unlocked key in a key session. By default ubuntu runs a key session, so one doesn't have to be set up.

You can also configure ssh to forward the key session to the target machine, allowing your user to log into other machines from the target machine in a chain.

Launching GUI apps

It's possible to use the SSH session to launch GUI apps on the target. For this we need to use a flag when accessing the machine. '-X'. This will forward the X session data to the host and allow the host to interact directly with the client's xorg GUI.

```
ssh -X student@192.168.50.14
gcalctool
```

This should load the gnome calculator. Without the -X flag the host will not have access to an xorg server (GUI) and attempting to load gcalctool will fail with an error that it can not connect to the display.

Kicking off Remote Users

A systems administrator can kick user off who are accessing a computer remotely. This can be a useful way to see who is accessing the computer and what they are running.

```
users
```

To see what users are logged in you can use the `users` command and to see what sessions are running use the `ps` and `grep` commands together. You are looking for sshd processes at pts such as student@pts/4, these can be listed using the following command:

```
ps ax | grep sshd:
```

Kicking a user is as easy as killing the session process that allows them to connect. Use the above command to find the process id and then use kill:

```
kill [pid-number]
```