Using SSH keys

WARNING: Incorrectly configuring ssh keys can leave your accounts vulnerable to attack and, more importantly, can provide attackers with a trivial means to transfer their attacks to other systems and organizations. Organizations you are affiliated with may require you to maintain certain standards of personal IT security to help limit the risk of their systems being compromised. Please consult their IT security policies and staff. Regardless of policy, you should consider it your responsibility to help avoid the compromise of any system you have been given access to by deploying safe IT security practices.

SSH Security:

- If you don't sufficiently understand ssh keys, DON'T use them. Use only passwords and use good ones - at least 9 characters long, a mixture of alphanumeric and non-alphanumeric characters and of mixed case. The password should be completely different to the password you use on any other system.
- NEVER copy a private key anywhere! The private key should remain in your .ssh directory on the system you generated it on and should be readable only by you.
- SSH key passphrases should be as secure as other passowrds.
- Never setup passphraseless ssh keys to allow unauthenticated login access between systems.

Background

Using SSH keys in the simplest way (with a passphrase) appears, superficially, no different to not using keys at all, i.e. just using a password. One difference is that the ssh private security token is at the source end, not at the remote destination - along with the authentication method, this means keys are much harder to brute-force attack. Another difference is that the keys are under your control and are your responsibility, not that of the system (local or remote). These differences allows keys to be used in more interesting and flexible ways than passwords and still provide security. There are two particular usage patterns that may be of interest to you in making the most of NCI systems:

- Using ssh agents and agent forwarding to move security all the way back to your desktop. This can greatly ease the burden and improve the security of ssh key usage (but, as with all security, still requires discipline and attention to detail).
- Using restricted command ssh keys *without passphrases* for limited functionality remote operations such as for transferring files.

It is the second of these we cover in detail below. Note that it is easy to unwittingly subvert SSH security if you are not careful when setting up restricted commands. Seek advice if you are unsure.

Using restricted commands for transferring files

NCI users often have chained workflows requiring automated transferring of files to or from remote systems or performing other operations on those systems. Since the scheduling of these operations is driven by a batch controlled workflow, relying on entering a password or passphrase at the time of the operation is not feasible. If the ssh keys are restricted to allow only those remote commands needed for the file transfers, then passphraseless keys can be used with some degree of security.

Usually, you are unaware of what commands are executed at the remote end when using a file transfer utility. Finding out what those commands are and configuring ssh to use them securely is, generally, non-trivial. Fortunately, the work has already been for `rsync`. On Linux system with `rsync` installed you will most likely find a file called something like /usr/share/doc/rsync/support/rrsync which is a "restricted rsync command target". Its certainly on Gadi.

The general idea is:

- create special purpose keys on the source system, assumed to be Gadi
- setup restricted command in authorized_keys on the target system (assumed to be your home institution).

Procedure to setup rrsync

1. Make sure perl and `rrsync` are installed on the destination host for the file transfers. You can just put `rrsync` in your personal bin directory there but make sure its executable:

   ```
   MyDesktop:~ > cp rrsync ~/bin
   MyDesktop:~ > chmod +x ~/bin/rrsync
   ```

2. Generate restricted command ssh keys on Gadi:

   ```
   [dbs900@gadi-login-07 ~]$:~/.ssh > ssh-keygen
   Generating public/private rsa key pair.
   Enter file in which to save the key (/home/900/dbs900/.ssh/id_rsa): /home/900/dbs900/.ssh
   /id_rsa_file_transfer
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in /home/900/dbs900/.ssh/id_rsa_file_transfer.
   Your public key has been saved in /home/900/dbs900/.ssh/id_rsa_file_transfer.pub.
   The key fingerprint is:  ....
   ```

   Yes, that was passphrase-less - just hit return when prompted for a passphrase.
3. Add the id_rsa_file_transfer.pub public key to the authorized_keys file on the file transfer target host but **only** with a restricted command prefix:

```
MyDesktop:~/.ssh > cat authorized_keys
...
from="gadi-dm*.nci.org.au,gadi*.nci.org.au",command="~/bin/rrsync /data/archive",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty,no-user-rc ssh-rsa AAAAB3N ... ynuw== dbs900@gadi-login-07.nci.org.au
...
```

Things to note:

- The entry has the command, other options and the public key (the bit after ssh-rsa) on the one line. Do not split them!
- Some of the restrictive options (no-..) may not be available on your system - look at the AUTHORIZED_KEYS FILE FORMAT section on the sshd man page on your system to see what is supported. The more restrictions, the better.
- The path under which you want data to be stored (/data/archive in this case) is given as an argument to `rrsync`.

4. On Gadi use something like:

```
rsync -vrlpt ./ExpDir/ -e "ssh -i $HOME/.ssh/id_rsa_file_transfer"  MyDesktop.myuni.edu.au:ExpDir
```

to archive a directory on Gadi to a directory on the remote system under the nominated archive directory there.