SSH with private/public key authentication

In this exercise we'll show how you can eliminate passwords by using ssh key authentication.

Choose the version of the exercises depending on what OS you are running on your laptop.

Remember: the `$` character before commands indicates that they are to be run as your normal login user, not as root.

For laptops running Windows

Download the following onto your desktop or into a downloads folder:

- putty.exe (you should already have this)
- psftp.exe
- pageant.exe
- puttygen.exe

from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html (Or you can try the installer bundle which gets them all)

Alternately you can use http://noc.ws.nsrc.org/downloads/ to download these files if the public download site is not available or too slow.

1. Generate an ssh public/private key pair

Double-click on `puttygen.exe`

At the bottom of the dialog box, under "Parameters":

- Make sure the type of key to generate is "SSH-2 RSA"
- Set the number of bits to 2048

Click on "Generate". Move the mouse randomly over the blank area until the progress bar reaches 100%

```
Key comment:        [Your Name <your@email.address>   ]
Key passphrase:     [chooose a passphrase             ]
Confirm passphrase: [choose same passphrase           ]
```

The passphrase is used to keep your private key encrypted on disk. It can be pretty much anything you want and as long as you want - including spaces - but if you forget it, your key becomes worthless. For now pick something that you will easily remember. You can change it at any time you want in the future.

Click “Save public key”. Give a filename of “id_rsa.pub” (please save files into the same directory as where the executables are)

Click “Save private key”. Give a filename of “id_rsa.ppk”

Use the mouse to highlight all the text in the box “Public key for pasting into OpenSSH authorized_keys file”, and copy it to the clipboard.

Exit puttygen.

NOTE: Key generation is a one-off exercise. The more you deploy your public key, the more work it to be if you were to lose it and have to start again with a new one. I suggest you keep a secure backup of it somewhere, e.g. on a CD-ROM or a safe USB key that you lock away.

2. Copy the PUBLIC key onto your Unix server

You have two ways of doing this.

Copy-paste

Use putty.exe to make a normal ssh connection to your host as the ‘sysadm’ user.

```
$ cat >>.ssh/authorized_keys
*** PASTE KEY FROM CLIPBOARD ***
*** If the cursor is still at the end of the line, hit Enter ***
*** hit ctrl-D ***
```

The key consists of one very long line, which looks like

```
ssh-rsa <lots of base64 data> <comment>
```

As a quick check that it hasn’t been corrupted, count the lines in the file:

```
$ wc -l .ssh/authorized_keys
1 .ssh/authorized_keys
```

If you don’t see “1”, then you’ll need to fix it (possibly with an editor, or else just rm the file and start again)

Now logout.

Alternative way (if you're having problems with copy-paste)

Double-click on psftp.exe. Open a connection to your server, and upload your public key:

```
psftp> open pcN.ws.nsrc.org
login as: sysadm
sysadm@pcN.ws.nsrc.org's password: <usual one>
Remote working directory is /home/sysadm
psftp> put id_rsa.pub
local:id_rsa.pub => remote:/home/sysadm/id_rsa.pub
psftp> quit
```

Unfortunately, this public key is not in the format which openssh requires, so now login again using putty.exe, and use the following command to convert it and put it in the right place.

```
$ ssh-keygen -i -f id_rsa.pub >>.ssh/authorized_keys
```

3. Login using your private key

Start putty.exe again. Enter the hostname as usual, but before clicking Open, browse in the left hand column to Connection > SSH > Auth

```
[-] Connection
     |
    [-] SSH
     |  |- Keyex
     |  |- Auth    <--- CLICK HERE
```

Next to "Private key for authentication", click Browse. Find your id_rsa.ppk file, open it, then click Open to start the connection.

You should be prompted for your username as before (sysadm), but then instead of being prompted for a password, you are asked for the passphrase for your private key. Enter it, and you should be logged in.

This is quite painful (both locating the private key and entering the passphrase), so as the final step of the exercise we're going automate it using an agent.

4. Use a passphrase agent

Run `pageant.exe`

It runs in the background, and adds an icon to your task tray (a PC with a black hat at a jaunty angle). You may need to select “Show hidden icons” to see it.

Right-click on the icon, and select “Add Key”. Browse to your id_rsa.ppk and open it. You will be prompted for the passphrase - enter it. (If you make a mistake, you’ll be prompted again until you get it right)

Now run putty.exe again, enter your hostname, click Open, and enter your username (sysadm). You should be logged in immediately, with no prompt for either a password or a passphrase!

Try logging in again. Also try using psftp.exe (when it runs, enter “open pcN.ws.nsrc.org” to start a connection). No passphrase is needed until you tell Pageant to forget the private key.

For laptops running Linux (or BSD or OSX)

1. Generate an ssh public/private key pair

```
$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sysadm/.ssh/id_rsa): <HIT ENTER>
Created directory '/home/sysadm/.ssh'.
Enter passphrase (empty for no passphrase): <CHOOSE PASSPHRASE>
Enter same passphrase again: <SAME PASSPHRASE>
Your identification has been saved in /home/sysadm/.ssh/id_rsa.
Your public key has been saved in /home/sysadm/.ssh/id_rsa.pub.
The key fingerprint is:
32:2b:e3:0e:14:fb:60:38:a6:e2:73:95:53:9d:a8:0f sysadm@pcN.ws.nsrc.org
```

The passphrase is used to keep your private key encrypted on disk. It can be pretty much anything you want and as long as you want - including spaces - but if you forget it, your key becomes worthless. For now pick something that you will easily remember. You can change it at any time you want in the future (using `ssh-keygen -p`)

NOTE: Key generation is a one-off exercise. The more you deploy your public key, the more work it to be if you were to lose it and have to start again with a new one. I suggest you keep a secure backup of it somewhere, e.g. on a CD-ROM that you lock away.

2. Copy the PUBLIC key onto your Unix server

The simplest way to copy the public key is with scp:

```
$ scp .ssh/id_rsa.pub sysadm@pcN.ws.nsrc.org:.ssh/authorized_keys
```

Note that .ssh/authorized_keys can contain multiple keys, one per line, so on a shared system you might want to append your key instead:

```
$ cat .ssh/id_rsa.pub | ssh sysadm@pcN.ws.nsrc.org 'cat >>.ssh/authorized_keys'
```

3. Login using your private key

Open an ssh connection to your server as normal:

```
$ ssh sysadm@pcN.ws.nsrc.org
```

This time, instead of being prompted for your password, you should be prompted for the passphrase on your private key. Enter it. You should be logged in.

4. Use a passphrase agent

Entering a passphrase every time you connect would be painful, but this isn't necessary if you have an agent which decrypts the private key and keeps it in memory.

If you are running under a modern graphical environment like Gnome, you probably already got a dialog box prompting you for a passphrase, and this means you're already running an agent. You should be able to logout and login to the remote server, without being prompted for your passphrase again.

To see what identities (decrypted private keys) your agent has in memory:

```
$ ssh-add -l
```

To forget all identities:

```
$ ssh-add -d
```

If you don't have an agent, then you can start a new subshell with ssh-agent as its parent:

```
$ ssh-agent bash
$ ssh-add
... prompted for your passphrase
$
```

Now the agent will handle future connections for you.

If you are running an older graphical environment, and you normally start X using `startx`, then start it using `ssh-agent startx` instead. Then type 'ssh-add' in an xterm.

Disable Password Access to your Machine

Connect Only With SSH Keys

Only do this exercise if you have successfully copied your public key to your machine and you are being prompted for your ssh private key passphrase when you log in and not your password.

Log in on your machine. Once logged in become the root user:

```
$ sudo bash
```

As the root user copy authorized_keys file to the directory /root/.ssh.

```
# mkdir /root/.ssh
# chmod 700 /root/.ssh
# cp /home/sysadm/.ssh/authorized_keys /root/.ssh/.
# chown root:root /root/.ssh/authorized_keys
# chmod 644 /root/.ssh/authorized_keys
```

Now log out of your machine and try to log back in, but this time as the "root" user, not as the "sysadm" user. If you are prompted for your ssh private key passphrase and not a password, then you are ready to disable password access to your machine.

Log in on your machine as the root user. Now we are going to edit the file /etc/ssh/sshd_config.

```
# vi /etc/ssh/sshd_config
```

Insie the file look for the following line:

```
#PasswordAuthentication yes
```

Just after this line add a line that says:

```
PasswordAuthentication no
```

Save and exit from the file. The reload the ssh server:

```
# service ssh reload
```

Before logging out we recommend you leave your current session open in case there are problems. You could lock yourself out of your machine. If you do let your instructor know. Now use ssh or putty on your laptop and try connecting to your server. You should get prompted for your ssh private key's passphrase and be able to log in. If you did, everthing is working. At this point all new users on your machine must use ssh keys to connect and not passwords. If you wish to verify this you can do the following:

```
# adduser testuser
```

Answer the on-screen prompts. Once the user has been created and you have given it a new password of your choosing, try to open a new ssh session to your machine as that user. You should either be rejected or your password attempts will have not affect (depends on the ssh server version).

To log in as this user you would need to copy your public ssh key to their account as we did for your sysadm user. This **greatly** enhances the security of your system.

Additional information [not part of exercises]

Agent forwarding

Using an agent, you can access across multiple ssh hops without having to copy your key or enter your passphrase anywhere.

If you enable "agent forwarding" when you login to host X, you can then login from X to Y without any prompting (assuming Y has your public key in authorized_keys). The request to authenticate is forwarded securely back along your original ssh session to the agent running on your workstation.

Under Unix:

```
$ ssh -o ForwardAgent=yes user@host
```

If you do this frequently, it's easier to configure it in `.ssh/hosts`

```
host foo
hostname foo.example.com
user sysadm
ForwardAgent yes
```

Then you only have to type `ssh foo` to get a connection with those options.

Advanced: X11 forwarding and port forwarding

ssh can securely carry arbitrary forwarded TCP connections and X11 graphics.

```
$ ssh -L8080:some.where:80 user@remote.host
... while ssh connection is open, a connection to 127.0.0.1 port 8080
... will be tunneled, and the far end will open a connection to
... some.where port 80

$ ssh -X user@remote.host
...
$ xclock   # graphical output redirected back through ssh tunnel
```