Configuring Key Based Authentication for SSH on Unix based computers

Ezra S. Frick

SSH (Secure Shell) is a computer program used to remotely log in to computers to execute instructions. This allows the user to remotely do anything that the user could normally be do while sitting in front of their own computer. This includes moving and altering files, running programs, and, if the user has the authority, management of other users on the machine. For this reason it is very important to keep the connection to it secure. SSH already does some things to protect itself. Most notably, it is a tunneled connection, meaning that the data about the session is encrypted before being sent unlike telnet witch transmitted everything in plain text.

It is however possible to make it more secure. One of the easiest ways to do this is to use key based authentication. This will use locally authenticated keys to verify users thus making users credentials much harder to steal and nearly impossible to brute force.

Requirements
A Unix based computer
A server running Linux
- you must have log-in credentials on this server
- you must have access to root privileges on this server (I will be using sudo throughout this instruction set.)

Demo's setup
Two virtual machines
Server running Debian 8.0.3
Workstation running Ubuntu 14.04

Logging in

1) Open a terminal window and enter the command *ssh user@hostname*

```
user01@demobox:~$ ssh user@192.168.0.102
```

If this is your first time logging in you will see a warning stating that the authenticity of the host you are connecting to cannot be established. Check the address and type yes to continue.

```
user01@demobox:~$ ssh user@192.168.0.102
The authenticity of host '192.168.0.102 (192.168.0.102)' can't be established.
ECDSA key fingerprint is 6a:e1:23:b6:00:00:e7:96:fa:a3:0f:75:31:7e:a8:06.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.102' (ECDSA) to the list of known hosts.
user@192.168.0.102's password:
```

2) You will then be prompted for a password for the remote machine. Enter yours at the prompt.

```
user01@demobox:~$ ssh user@192.168.0.102
The authenticity of host '192.168.0.102 (192.168.0.102)' can't be established.
ECDSA key fingerprint is 6a:e1:23:b6:00:00:e7:96:fa:a3:0f:75:31:7e:a8:06.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.102' (ECDSA) to the list of known hosts.
user@192.168.0.102's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Feb 28 11:45:05 2016
user@demoserv:~$
```

Making the key

1) Open a second terminal window and enter the command *ssh-keygen*.

```
user01@demobox:~$ ssh-keygen
```

A key pair will be generated and you will be prompted for a location to save them to. Accept the default by pressing enter.

```
user01@demobox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user01/.ssh/id_rsa):
```

You will now be prompted for a password to authenticate the key. [**Note:** This should be a different password than your log-in password.] You will be asked to reenter your password for verification.

```
user01@demobox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user01/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

The key pair will then be saved in the location specified and some randomart will generate.

```
user01@demobox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user01/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user01/.ssh/id_rsa.
Your public key has been saved in /home/user01/.ssh/id_rsa.pub.
The key fingerprint is:
bf:57:94:ef:f6:65:90:50:4c:a8:ae:56:a7:57:57:ca user01@demobox
The key's randomart image is:
+--[ RSA 2048]----+
|             +o  |
|            ...  |
|           ..  . |
|          .  .o..|
|        S.   ooo.|
|         .o . E.o|
|         o.o o oo|
|        o ..o  .+|
|       .  .o   .o|
+-----------------+
user01@demobox:~$
```

Copying the public key to the remote server

1) Copy the public key to the remote server with the command *spc .ssh/id_rsa.pub user@hostname:*.

```
user01@demobox:~$ scp .ssh/id_rsa.pub user@192.168.0.102:.
```

You will then be prompted for your password. Enter it at the prompt. This will copy id_rsa.pub into your home folder on the remote server.

```
user01@demobox:~$ scp .ssh/id_rsa.pub user@192.168.0.102:.
user@192.168.0.102's password:
id_rsa.pub                                    100%  396     0.4KB/s   00:00
user01@demobox:~$
```

2) Switch back to the first terminal window. You will have landed in your home directory on the remote machine. Here, make a sub-directory called .ssh using the command *mkdir .ssh*

```
user@demoserv:~$ mkdir .ssh
```

3) It will be necessary to change the default permissions given to this directory so that only you have access to it. Enter the command *chmod 0700 .ssh*

```
user@demoserv:~$ chmod 0700 .ssh/
```

4) Now move id_rsa.pub into .ssh using *mv id_rsa.pub .ssh/authorized_keys*. This will also rename the file to authorized_keys.

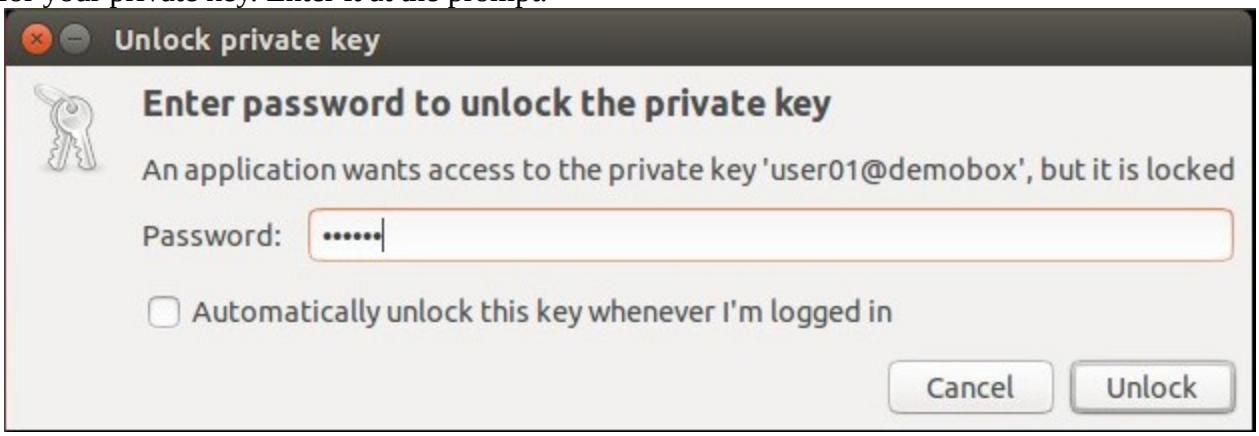
```
user@demoserv:~$ mv id_rsa.pub .ssh/authorized_keys
```

5) It is now necessary to change the file permissions on the authorized_keys file. Enter the command *chmod 0600*

```
user@demoserv:~$ chmod 0600 .ssh/authorized_keys
```

Testing the keys

1) Log off of the server and reenter *ssh user@hostname*. This time you will be asked for the password for your private key. Enter it at the prompt.



[**Note:** Not all systems will open a second window. Many will prompt for the password on the command line.]

If you are not logged on then review the previous steps and ensure that they have been done correctly.

Disabling Log-in via password (optional)

[**WARNING: Do Not Attempt This Step Until You Have Verified That the Keys Are Working Properly. Failure to do so May Result in You Being Lock Out of Your Server.**]

1) On the remote machine, open the file /etc/ssh/sshd_config with a text editor. This file is read only so you will need root permissions to edit it. [**WARNING: This file sets the configuration for the ssh server on the remote machine. Improper editing of this file could cause you to be locked out of your server.**]

```
user@demoserv:~$ sudo vim /etc/ssh/sshd_config
```

2) At about line 52 delete the pound sign in front of PasswordAuthentication and change the *yes* to a *no*.

```
51 # Change to no to disable tunnelled clear text passwords
52 #PasswordAuthentication yes
```

```
51 # Change to no to disable tunnelled clear text passwords
52 PasswordAuthentication no
```

3) Save your changes and exit.

```
user@demoserv:~$ sudo service ssh restart
```

4) It is now necessary to restart the ssh server to apply the new rules for authentication. Enter the command *sudo service ssh restart*. [**Note:** Some distributions will require *sshd* in the previous command.]

5) Congratulations you have now configured key based authentication for ssh.

SSH is a powerful tool for anyone managing a Unix based server. It allows for easy remote management and gives you the ability to easily manage multiple servers from a single machine. However, with great power comes the need for great security. Key based authentication is just one of many tools that can insure that this powerful tool doesn't fall into the wrong hands. Keep your servers yours and keep them doing great things.