

# Securing Your OpenSSH Server

(Document v1 – 13 August 2008)

## Introduction

Since brute force attacks against SSH servers are widely carried out, especially by means of automated scripts, you need to harden your SSH server. Default OpenSSH Configuration has many security vulnerabilities that must be patched by modifying the configuration. This document explains the steps you can take to secure your OpenSSH installation. Sri Lanka CERT recommend you carry out all the suggested steps in this document to provide defense-in-depth for the SSH server.

## Test System

All configurations and commands used in this document have been tested in a system running OpenSSH on Ubuntu Dapper 6.06 (Desktop). The same configuration steps and commands should work in other flavours/versions of Linux with minor modifications.

## Save Your Existing SSH Configuration

First you need to save the existing *sshd\_config* file just in case if you need to replace that in the future. Type:

```
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original
```

## Restarting SSH After Configuration Changes

For the configuration changes suggested below to be effective you have to restart the SSH demon after you make changes to the configuration file. You can restart SSH by typing:

```
/etc/init.d/ssh restart
```

## Editing the SSH Configuration File

To make changes to your SSH demon you have to edit the ssh configuration file. By default this file is: */etc/ssh/sshd\_config*. (Do not confuse this with *ssh\_config*, which is the SSH Client configuration file). In order to edit the file open it in your favourite text editor. (I used vim)

### **Step 1: Allow Less Time for Users to Login**

By default, OpenSSH server allows for 120 seconds (two minutes) from the time the login prompt is displayed until a user must begin the login process before the connection is terminated by sshd. You should reduce this time limit (lets say to 20 secs). To achieve this locate and change the following line in *sshd\_config*:

*LoginGraceTime 20*

### **Step 2: Add a Login Banner to Warn Intruders**

You should add a login banner to warn off any intruders. This will also protect you legally if you intend to log the user activity.

In order to add a login banner first uncomment the line (by removing the # in front):

*#Banner /etc/issue.net*

Or add a new line

*Banner /etc/issue.net*

To create the banner text, create a file:

*vim /etc/issue*

Enter banner text of your preference such as “*This computer is accessible only for authorized users. Any unauthorized use or such attempt will result in severe criminal penalties to the extent possible by applicable law. If you are not authorized to use this computer log off immediately*” and save the file.

Then create a symbolic link to the banner text file:

*ln -s /etc/issue /etc/issue.net*

### **Step 3: Only Allow Selected Users to Login via SSH**

You should only allow only necessary users to login to SSH server. To do this, include the following line:

*AllowUsers user1 user2 user3*

User1, user2 etc are the usernames of users who are authorized to login via SSH.

## Step 4: Disable root Login via SSH

You should disable root login via SSH. Users should login using their individual login and use *su* command to get privileged access. In order to disable root access change *PermitRootLogin* parameter to *no*.

*PermitRootLogin no*

## Step 5: Change Default SSH Listening Port

Many SSH password brute force tools will try to connect to default SSH listening port (port 22). Therefore, you should change the default listening port to something else. (Port 2223 in this example). To do this, add or change the following line in *sshd\_config*:

*Port 2223*

## Step 6: Use Public Key Authentication

The best way to defeat SSH brute force attacks is to disable password authentication and use public key authentication instead. Following steps will illustrate how to achieve this.

### 6.1 Generate a Key Pair

First generate a RSA key pair. Ideally this should be done in the SSH client machine. Alternatively you can create this in any pc with SSH installed and later copy the keys into the SSH client PC. The client PC can be running either Linux or Windows. The two cases will be described separately.

#### 6.1a Generating a Key Pair (Linux)

In a Linux box with SSH client type the following to generate your RSA key pair.

```
ssh-keygen -t rsa -b 4096
```

*-t* parameter specifies the type of key and in this example it is RSA. *-b* parameter describes the private key length. Longer private keys will be more secure.

You will be prompted for the private key file name and path. Type a path you desire or press return to accept the defaults. When prompted enter a strong passphrase so that your private key will be encrypted using symmetric encryption. (If you forget the passphrase later your private key will be unrecoverable)

Two keys will be generated here:

Private key named *id\_rsa*

Public key named *id\_rsa.pub*

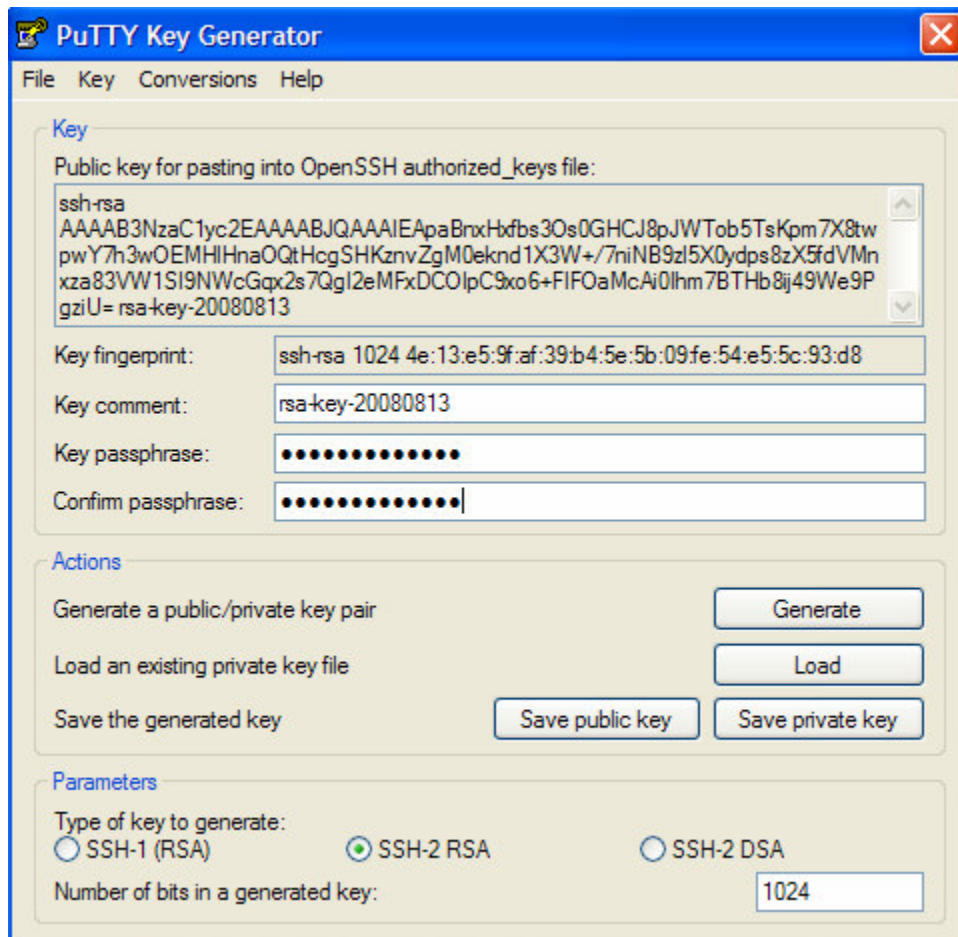
The private key should be kept encrypted (through passphrase) in the SSH client PC. Never give this to anyone else or copy it to another machine. You can keep a copy of your public key with you as well. However, you must copy your public key to the SSH server.

### 6.1b Generating a Key Pair (Windows)

If you are using Windows most probably you are using PuTTY as the SSH client. There is a utility called PuTTYgen to generate key pairs for use with PuTTY. You can download both PuTTY and PuTTYgen from:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

To generate a key pair just run PuTTYgen and click “Generate”.



Once keys are generated enter a passphrase and use “Save public key” and “Save private key” options to save them to a preferred location.

Alternatively, you can generate the key pairs in a Linux box as in 6.1a and still use that with PuTTY. However, you should first load it to PuTTYgen using the “Load” option (It will prompt for your passphrase) and save the key pair in a format readable by PuTTY (As a .ppk file)

## 6.2 Copying Your Public Key to Server

Login to the SSH server and create a directory to store your SSH public key if such directory does not exist.

```
mkdir /home/username/.ssh
```

Where *username* is your username in OpenSSH server

Copy your public key from the client PC to the above directory created in the SSH server (i.e., */home/username/.ssh*).

Go to the */home/username/.ssh* directory and concatenate the public key to a file named *authorized\_keys*:

```
cd /home/username/.ssh  
cat id_rsa.pub >> authorized_keys
```

If no *authorized\_keys* file is available it will be created by the *cat* command.

Modify permissions to the file (Only user should have access to that)

```
chmod 700 authorized_keys
```

## 6.3 Change the SSH Configuration to Use Public Key Authentication

Open the *sshd\_config* file:

```
vim /etc/ssh/sshd_config
```

Make sure that following lines are present and uncommented (If not add them):

```
RSAAuthentication yes  
PubkeyAuthentication yes  
AuthorizedKeysFile %h/.ssh/authorized_keys
```

Save the *sshd\_config* file and exit from your text editor.

Restart the SSH demon using the following command:

```
/etc/init.d/ssh restart
```

Now you should be able to login using public key authentication to the SSH server.

#### 6.4a Login Using Public Key Authentication (Linux)

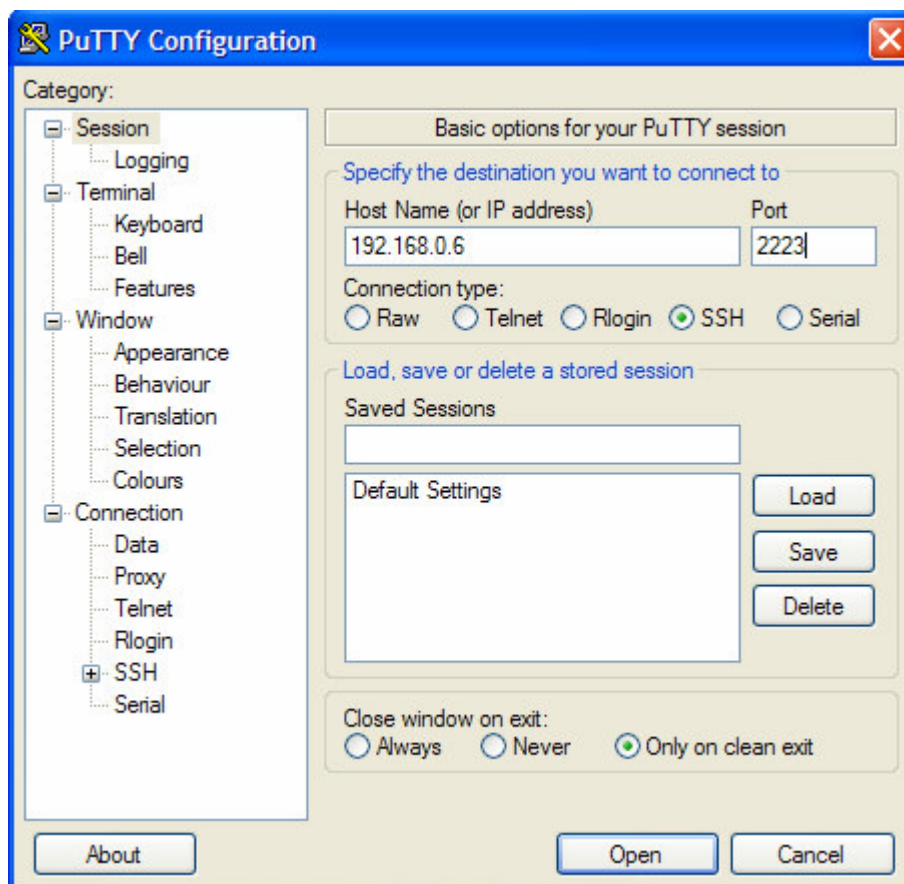
If you are using a Linux box as the SSH client simply type:

```
ssh username@server_ip -p 2223
```

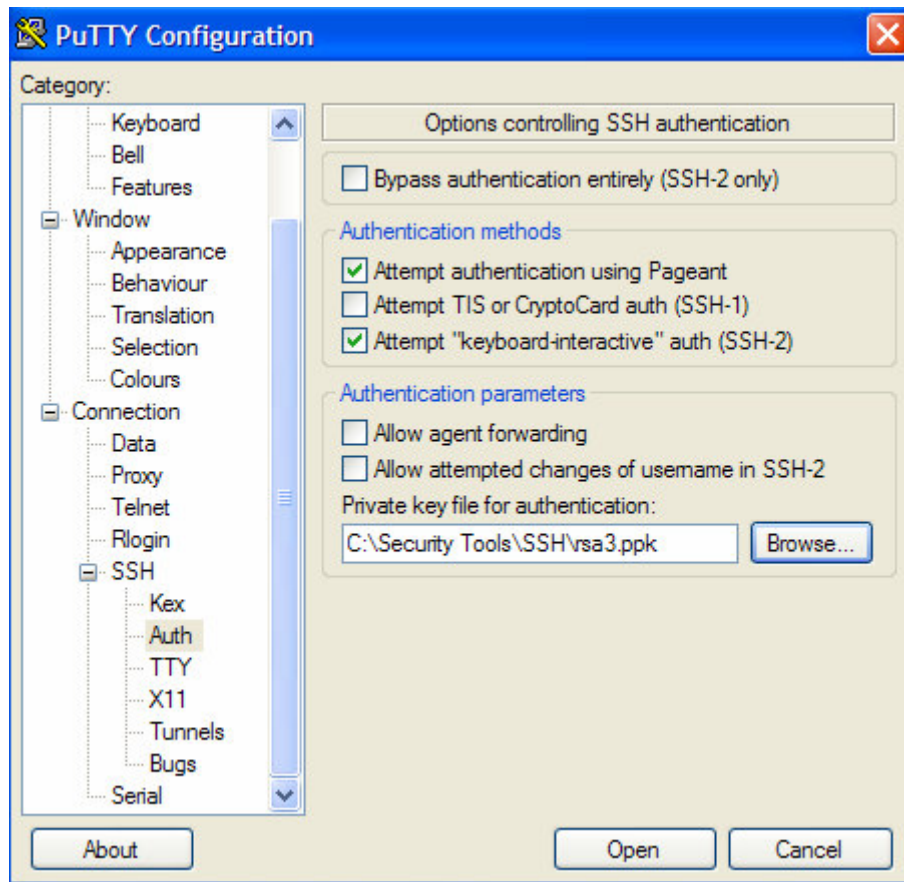
Where *username* is your username in SSH server. `-p` option specifies the new port number which is 2223. SSH Client will prompt you to enter your passphrase and once that is done you will be logged in.

#### 6.4b Login Using Public Key Authentication (Windows)

You can use PuTTY as the SSH client in Windows. First open PuTTY and enter the SSH Server IP, Port and select SSH.



Then expand “SSH” in the left pane and select “Auth”. Click “Browse”, select and load your private key file to PuTTY. Finally click “Open” to initiate the SSH session. You will be prompted for the passphrase of the private key before logon.



### Step 7: Disable Password Authentication

Although you enabled public key authentication in Step-6 SSH Server will still use password authentication if primary authentication method fails. Once, Public key authentication is up and running you can disable password authentication by changing the *PasswordAuthentication* line in *sshd\_config* to *no*:

*PasswordAuthentication no*

### Step 8: Use Only SSH Version 2

SSH Version 1 has many vulnerabilities and use of SSH Version 2 is recommended. Usually, you don't have to worry about this since new SSH installations will use version 2 by default. Yet, just to make sure check whether the following line is present in *sshd\_config* and if not insert it.

*Protocol 2*

## Step 9: Use Iptables to Filter SSH Traffic

You can use Iptables to allow SSH traffic only from required hosts (e.g. Only allow administrators laptop to access a server via SSH). Assuming only host 192.168.0.10 needs to login via SSH type the following commands to add simple rules to the INPUT chain:

```
iptables -A INPUT -p tcp -m state --state NEW --source 192.168.0.10 --dport 2223 -j ACCEPT
```

Above line will accept SSH connections coming from 192.168.0.10.

```
iptables -A INPUT -p tcp --dport 2223 -j DROP
```

Above line will drop all other SSH connections.

## Step 10: Use TCP Wrappers to Restrict SSH Access

In addition to Iptables TCP Wrappers can also be used to restrict access to SSH demon (If this feature is available in your Linux distribution). You need to edit */etc/hosts.allow* and */etc/hosts.deny* files to achieve this.

First edit the etc/hosts.allow file and add the following line:

```
sshd: 192.168.0.10/255.255.255.255
```

The above line allows host 192.168.0.10 to connect to the SSH server. The line can be modified if you need to allow a network range.

Then edit the etc/hosts.deny file and add the following line:

```
sshd: ALL
```

The above line will deny all other access to SSH demon.

## Step 11: Disable TCP/X11 Forwarding

TCP forwarding allows tunneling of traffic to other services via SSH, where X11 allows graphical programs (Such as Linux Terminal Server) to be tunneled through SSH. If these services are not required you must block them by setting the appropriate option in the sshd\_config to no as shown below:

```
AllowTcpForwarding no  
X11Forwarding no
```

## Disclaimer of Liability

*With respect to the advise/solutions offered in the above communication, neither Sri Lanka CERT nor any of its employees or consultants, make any warranty, express or implied, including the warranty of merchantability and fitness for a particular purpose, or assume any liability or responsibility for the accuracy, completeness, or usefulness of the information contained in this communication.*