By Daniel Boros
Published: 2008-03-27 11:16

How To Set Up SSH With Public-Key Authentication On Debian EtchPreliminary Notes

This mini-howto explains how to set up an SSH server on Debian Etch with public-key authorization (and optionally with disabled password logins). SSH is a great tool to control Linux-based computers remotely. It's safe and secure.

There's no warranty that it'll work for you. All of these settings are applicable for Debian and -like systems! There may be slightly changes on other systems as well.

Installing SSH On The Server

First, we install the SSH on our server. We can do that with this command: (Note that you must be *root* to do that!)

```
apt-get install ssh
```

Preparations On Our Client (Desktop) System

Second, we take some preparations on our desktop machine. This PC will be used to connect the server. So, the SSH-server has been installed on a **different** machine.On your desktop machine, we install the ssh client (which we use to connect the server).Note that installing programs requires *root* privilege! If you're not logged in as *root*, please log in! (*su root* then type your password.) Then install the client:

```
apt-get install openssh-client
```

Switch back to your normal user (not *root*, respectively).Then type these commands in order:

```

mkdir ~/.ssh
```

```
chmod 700 ~/.ssh

cd ~/.ssh
```

We generate our key-pair, a public-key and a private-key. The public-key will be placed on the server, and you will log in with your private-key. When asked, type your passphrase (it'll be needed for future logins, so remember it!):

```
ssh-keygen -t rsa -C "A comment... usually an email is enough here..."
```

Then we copy the public key (which we've generated just before) to our (remote) server. The remoteuser should not be `root`! Choose the default non-root user as remoteuser. (Note the colon at the end of the line! It's important.)

```
scp -p id_rsa.pub remoteuser@remotehost:
```

Then we log in with SSH, and we copy the public key to its right place:

```
ssh remoteuser@remotehost

mkdir ~/.ssh

chmod 700 ~/.ssh

cat id_rsa.pub >> ~/.ssh/authorized_keys

chmod 600 ~/.ssh/authorized_keys

mv id_rsa.pub ~/.ssh

logout
```

We have to delete the public key on the desktop, because otherwise the SSH client doesn't allow us to log in to the server. So, type this command:

```
rm id_rsa.pub
```

And then we log back:

```
ssh remoteuser@remotehost
```

If we've done everything precisely as detailed above, then you'll be asked for the passphrase. Type it, then you are in and have a fairly safe SSH-environment!

Disabling Password Authentication

Disabling it is a good way to have a safer SSH-installation. Then you can log in **only with** a key-pair, so be careful not to lose it! It's purely optional but safe to activate! But before doing it, please make sure that key-based authentication is working out-of-the-box.Sit down in front the server (so don't log in remotely as we have to restart the SSH later...) and type these commands manually as `root`:

```
cd /etc/ssh

cp sshd_config sshd_config.orig

nano sshd_config
```

You will have the nano text-editor on screen open with the main SSH configuration file. Change these lines (don't bother if any of these lines have a '#' mark at the beginning; if they have, just delete the hashmark as well):

```
PermitRootLogin  yes
PasswordAuthentication yes
UsePAM   yes
```

To these:

```
PermitRootLogin  no
PasswordAuthentication no
UsePAM   no
```

Then save the file with `Ctrl + O`, and restart the SSH server:

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

Be careful: if you disable password authentication, then you won't be able to log in with passwords! Only key-based authentication will be available!

Useful Links:

http://www.openssh.org
http://en.wikipedia.org/wiki/Secure_Shell