SSH configuration for a better security

1.0 Introduction

Secure Shell or SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices. Common uses of SSH includes:

- Login to a shell on a remote host (alternative for telnet and rlogin)
- Executing a single command on a remote host (alternative for rsh)
- Secure file transfer (SCP and SFTP)
- Port forwarding and tunneling
- Mounting a directory on a remote server as a filesystem (SSHFS)

The standard TCP port 22 has been assigned for contacting SSH servers.

An SSH client program is typically used for establishing connections to an SSH daemon accepting remote connections. Both are commonly present on most modern operating systems, including Mac OS X, Linux, FreeBSD, Solaris and OpenVMS. Proprietary, freeware and open source versions of various levels of complexity and completeness exist.

2.0 Security Issues

Since SSH-1 has inherent design flaws that make it vulnerable (e.g., man-in-the-middle attacks), it is now generally considered obsolete and should be avoided by explicitly disabling fallback to SSH-1. While most modern servers and clients support SSH-2, some organizations still use software with no support for SSH-2, and thus SSH-1 cannot always be avoided.

In all versions of SSH, it is important to verify unknown public keys before accepting them as valid. Accepting an attacker's public key as a valid public key has the effect of disclosing the transmitted password and allowing man-in-the-middle attacks.

In a normal server situation nowadays, most successful break-in are result from a successful brute-force attacks.

Illustration 1: Brute Force Attack

Illustration 1 shows a brute force the output from the /var/log/authlog in one of the OpenBSD Server (the hacking hostname has been obfuscated). You can see that this is a brute force attack trying to crack the "root" password. The time between the attack is just a second or less, which would be too quick for a normal human. This is most lightly to be an automated attack from a brute-forcing script.

3.0 Tips and tricks

Most administrators tend to install an SSH server and leave it at its default settings, a typical intruders/attacker may take advantages on the default settings such as default port and root login. Below are few of the steps that you might want to take note in securing your SSH servers.

3.1 Change the default ssh port number

By default, ssh listen to port 22 for connections. Attackers will use a port scanner software (such as nmap) to scans for an open port. Normally these port scanners do not scan higher ports.

To change the default port, you will need to open the configuration file using your favorite text editor (vi,nano) located in **/etc/ssh/sshd_config** and look for a line that says **Port 22**

Illustration 2: sshd_config file that mention Port 22

Change the port number to something like **Port 10000 #use your own** and restart the ssh server by typing **/etc/init.d/ssh restart** *note: you may need to be root in order to change and restart the ssh service. By changing the default port, you now can connect to the server by typing the following command: ssh –p username@server

3.2 Allow only specific users to login:

Not every users need to be able to gain ssh service on your server. You can specify which user can connect to the server by changing the configuration file in **/etc/ssh/sshd_config**
For example, if you only allow user "john" to connect to the server, add in the line below:
AllowUsers john

3.3. Do not allow root ssh login

It is always wise not to allow root login because common brute-force attack is using username root. Change or put in the line in your **/etc/ssh/sshd_config**
PermitRootLogin no

3.3 Disable keyboard interactive login

This setting gives you a decent protection against automated brute-forcing password attempts. But the downside is that the user will need to take some time to create encryption keys before they can log in.

First thing that you need to do is to create a key pair. You can do that by typing the following command in your Linux terminal:

$ ssh-keygen –v –t rsa

If you notice the command is followed with two option which is –v and –t.

The –v command is for verbose. It is not really a necessary option to put but just in case if you like to see what is going on.

The second is –t for setting up the type of key to be generated. You can use either RSA or DSA for the encryption.

After you have entered the above command, you will be prompted for the location to save the file. The default location is in the folder /.ssh/id_rsa/ or /.ssh/id_dsa depending on what kind of encryption did you use.

Just hit enter to save if to your default location. You will then be prompted for a passphrase. Just hit enter for an empty passphrase. This way, you can disable the interactive keyboard login.

You should be able to see something like this:

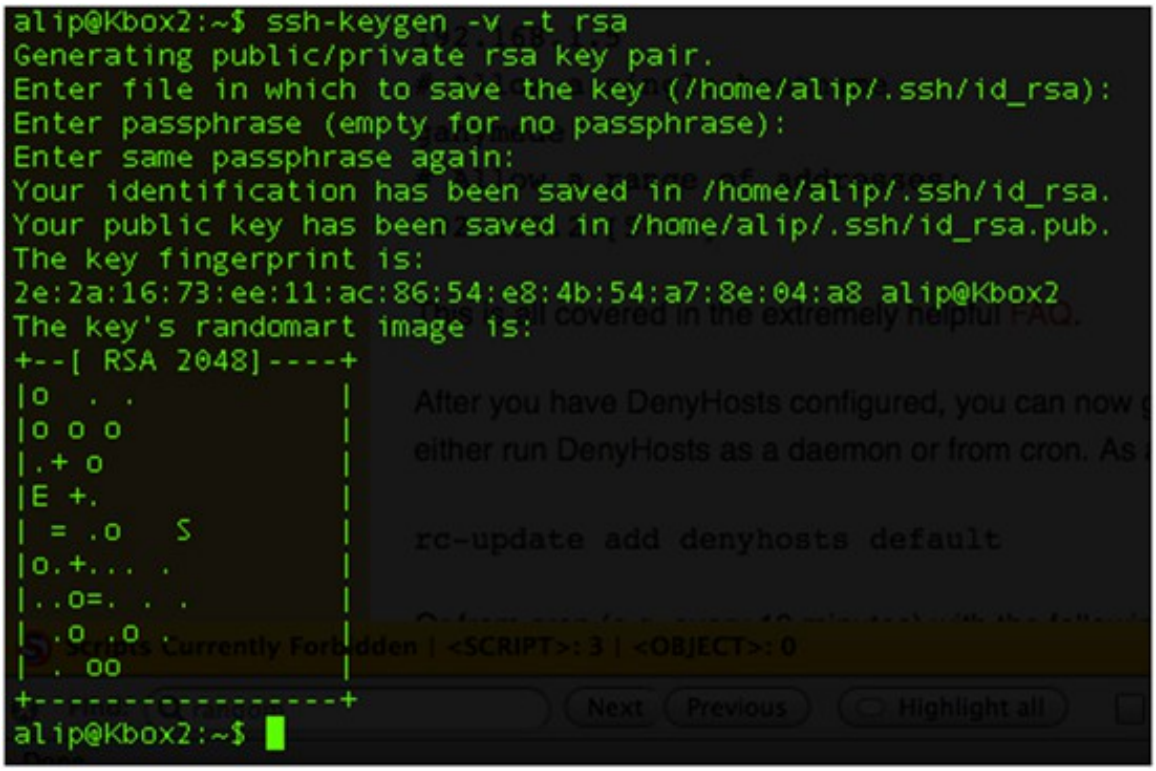

Illustration 3: Creating public/private key pair

The next step is to copy your generated public key to you .ssh/authorized_keys file into the server that you want to access. To do that, just type in the command below:
$ cat id_rsa.pub | ssh username@server "cat >> .ssh/authorized_keys

To enable key-based logins you need to tweak your sshd_config file and enable it with two lines below:

PubkeyAuthentication yes

AuthorizedKeysFile .ssh/authorized_keys

It is important that you confirm that you can log in without a password. To do that , just try the command below :

$ ssh server

You should be able to login to your ssh server like normal. You can now disable the keyboard logins with modifying/adding the two lines below to your /etc/ssh/sshd_config :

PasswordAuthentication no

ChallengeResponseAuthentication no

3.4 Blacklisting and Whitelisting with DenyHosts

DenyHosts is a tools that looks for bad login or attempts from hosts then add them to a blacklist. More information on the DenyHosts features can be found here:
http://denyhosts.sourceforge.net/features.html
To install DenyHosts on Ubuntu server is very easy. You can do that by typing

$ sudo apt-get install denyhosts

DenyHosts acts as a dynamic blocker for SSH and other services. It relies on the /etc/hosts.deny and hosts.allow.

The file /etc/hosts.deny is where you need to list all the ip that you want to block from accessing your ssh server and on the other hand the /etc/hosts.allow is the list that you want to allow.

This article will show you how to allow only specific ip or range of ip in the network to access your ssh server. You can do that by editing your

/etc/hosts.deny and add in the following lines

ALL:ALL

/etc/hosts.allow and add in the following lines

ALL: 192.168.1.0/24

This will only allow the range of ip 192.168.1.* to access your ssh server.

4.0 Conclusion

SSH server is actually pretty much secure as it is, but of course you can always tweak a few things to make it much more safer than the most commonly used attack vector.