**SSH**

## Introduction

This section of the Ubuntu Server Guide introduces a powerful collection of tools for the remote control of, and transfer of data between, networked computers called OpenSSH. You will also learn about some of the configuration settings possible with the OpenSSH server application and how to change them on your Ubuntu system.

OpenSSH is a freely available version of the Secure Shell (SSH) protocol family of tools for remotely controlling, or transferring files between, computers. Traditional tools used to accomplish these functions, such as telnet or rcp, are insecure and transmit the user's password in cleartext when used. OpenSSH provides a server daemon and client tools to facilitate secure, encrypted remote control and file transfer operations, effectively replacing the legacy tools.

The OpenSSH server component, sshd, listens continuously for client connections from any of the client tools. When a connection request occurs, sshd sets up the correct connection depending on the type of client tool connecting. For example, if the remote computer is connecting with the ssh client application, the OpenSSH server sets up a remote control session after authentication. If a remote user connects to an OpenSSH server with scp, the OpenSSH server daemon initiates a secure copy of files between the server and client after authentication. OpenSSH can use many authentication methods, including plain password, public key, and Kerberos tickets.

## Installation

Installation of the OpenSSH client and server applications is simple. To install the OpenSSH

Ubuntu: sudo apt-get install openssh-client

Redhat: yum install openssh

Suse: zypper install openssh

The openssh-server package can also be selected to install during the Server Edition installation process.

## Configuration

You may configure the default behavior of the OpenSSH server application, sshd, by editing the file /etc/ssh/sshd\_config. For information about the configuration directives used in this file, you may view the appropriate manual page with the following command, issued at a terminal prompt:

man sshd\_config

There are many directives in the sshd configuration file controlling such things as communication settings, and authentication modes. The following are examples of configuration directives that can be changed by editing the /etc/ssh/sshd\_config file.

Prior to editing the configuration file, you should make a copy of the original file and protect it from writing so you will have the original settings as a reference and to reuse as necessary.

Copy the /etc/ssh/sshd\_config file and protect it from writing with the following commands, issued at a terminal prompt:

sudo cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.original

sudo chmod a-w /etc/ssh/sshd\_config.original

The following are examples of configuration directives you may change:

* To set your OpenSSH to listen on TCP port 2222 instead of the default TCP port 22, change the Port directive as such:

Port 2222

* To have sshd allow public key-based login credentials, simply add or modify the line:

PubkeyAuthentication yes

If the line is already present, then ensure it is not commented out.

* To make your OpenSSH server display the contents of the /etc/issue.net file as a pre-login banner, simply add or modify the line:

Banner /etc/issue.net

In the /etc/ssh/sshd\_config file.

After making changes to the /etc/ssh/sshd\_config file, save the file, and restart the sshd server application to effect the changes using the following command at a terminal prompt:

sudo service ssh restart

Many other configuration directives for sshd are available to change the server application's behavior to fit your needs. Be advised, however, if your only method of access to a server is ssh, and you make a mistake in configuring sshd via the /etc/ssh/sshd\_config file, you may find you are locked out of the server upon restarting it. Additionally, if an incorrect configuration directive is supplied, the sshd server may refuse to start, so be extra careful when editing this file on a remote server.

## SSH Keys

SSH keys allow authentication between two hosts without the need of a password. SSH key authentication uses two keys, a private key and a public key.

To generate the keys, from a terminal prompt enter:

ssh-keygen -t rsa

This will generate the keys using the RSA Algorithm. During the process you will be prompted for a password. Simply hit Enter when prompted to create the key.

By default the public key is saved in the file ~/.ssh/id\_rsa.pub, while ~/.ssh/id\_rsa is the private key. Now copy the id\_rsa.pub file to the remote host and append it to ~/.ssh/authorized\_keys by entering:

ssh-copy-id username@remotehost

Finally, double check the permissions on the authorized\_keys file, only the authenticated user should have read and write permissions. If the permissions are not correct change them by:

chmod 600 .ssh/authorized\_keys

You should now be able to SSH to the host without being prompted for a password.

**SSH Agent**

The SSH agent handles signing of authentication data for you. When authenticating to a server, you are required to sign some data using your private key, to prove that you are, well, you.

As a security measure most people sensibly protect their private keys with a pass phrase, so any authentication attempt would require you to enter this passphrase. This can be undesirable, so the ssh agent caches they key for you and you only need to enter the password once, when the agent wants to decrypt it (and often not even that, as the ssh agent can be integrated with pam, which many distros do).

The SSH agent never hands these keys to client programs, but merely presents a socket over which clients can send it data and over which it responds with signed data. A side benefit of this is that you can use your private key even with programs you don't fully trust.

Another benefit of the SSH agent is that it can be forwarded over SSH. So when you ssh to host A, while forwarding your agent, you can then ssh from A to another host B without needing your key present (not even in encrypted form) on host A.

More info about ssh/ssh-keys/ssh-agent:

<https://www.openssh.com/manual.html>

What is important from the above link:

* [ssh(1)](http://man.openbsd.org/ssh) - The basic rlogin/rsh-like client program
* [sshd(8)](http://man.openbsd.org/sshd) - The daemon that permits you to log in
* [ssh\_config(5)](http://man.openbsd.org/ssh_config) - The client configuration file
* [sshd\_config(5)](http://man.openbsd.org/sshd_config) - The daemon configuration file
* [ssh-agent(1)](http://man.openbsd.org/ssh-agent) - An authentication agent that can store private keys
* [ssh-add(1)](http://man.openbsd.org/ssh-add) - Tool which adds keys to in the above agent
* [scp(1)](http://man.openbsd.org/scp) - File copy program that acts like rcp
* [ssh-keygen(1)](http://man.openbsd.org/ssh-keygen) - Key generation tool

Extra documentation:

1. <https://support.suso.com/supki/SSH_Tutorial_for_Linux>
2. <https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process>
3. Ssh-agent: <http://mah.everybody.org/docs/ssh>