



Secure Shell

*This material is based on work supported by the
National Science Foundation under Grant No. 0802551*



*Any opinions, findings, and conclusions or recommendations expressed in this material are those of
the author (s) and do not necessarily reflect the views of the National Science Foundation*

Lesson Overview

Secure shell refers to a protocol or method that allows one computer to access another computer across a network in a very secure manner.

In this lesson, you will explore secure encrypted Internet sessions, and you will learn how to configure OpenSSH—a favorite tool used for Secure Shell sessions. Additionally, you will explore the different options available to you through SSH connections.

This lesson is part of the Linux administration course. Secure Shell is an important topic because Linux administrators and other employees often need to access business computers remotely, and they must be trained to establish such access in a secure manner to prevent or minimize data loss, network vulnerabilities, or infiltration by unauthorized users.



Student Expectations

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives.

Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.



Objective

Given a need to secure Internet transmissions, the student will be able to describe the importance of tools such as SSH and configure OpenSSH for secure transmission as per industry standards.



Outline

In this lesson, you will explore:

- Important terms related to SSH
- Installing OpenSSH on Ubuntu
- Installing OpenSSH on Fedora
- Configuring SSH server
- Creating public and private keys
- SSH Tunneling



Introduction

The company for which you work as a Linux administrator is small. So, despite your spouse's protests, you take your company's laptop with you on vacation to be able to login to the company's network remotely and reset passwords once weekly.

One of your managers expressed concern about you accessing the network from your laptop because he read someplace that hackers can see all the text transmitted on the Internet, including the passwords you are going to reset.

Additionally, your office uses a text-based application to manage payroll and customer service responses. One manager from each department needs to be able to access the text-based app while on vacation in a secure manner.

With these concerns in mind, you begin researching options for secure remote access to the network.

Select **PLAY** below to view an introductory video.

View Video
VideoLesson7Introduction(C2L7
S14).mp4



Introduction (Contd)

As you research your situation, you focus on the following questions:

1. Why must information be encrypted as it moves across the Internet?
2. What software is available to do this encryption?
3. How does one configure OpenSSH to be both a server and terminal program?
4. What other functions can be completed using SSH besides direct connections to another Unix based machine?
5. What is meant by SSH tunneling?

After doing your research, you return to your manager and addressed his concerns regarding remote access and security.

Note: When you complete this lesson, you should be able to answer the five question noted above.



Definitions: Plain Text

When you send data across the Internet using words, letters, symbols, and numbers, you are using plain text.

If an intruder were to observe your activity and capture the data stream or packets you are transmitting, the text you are typing in your email or terminal window would be displayed to the intruder.

Consequently, using plaintext to transmit data is very insecure because passwords, financial data, health information and other personal data may be exposed to anyone who views the data packets. In the early days of credit card processing, plaintext transmissions allowed criminals to steal credit card information.

The alternative to plaintext is encryption .



Required Reading:

- [What is plaintext?](#)

Definitions: Data Encryption

Data encryption is the process of rearranging or encoding plaintext data using an algorithm or code so that the data is unreadable or inaccessible without the code. The process of coding plain text data is called *data encryption* or *encryption*, and the process of reverting coded data to its original plain text form is called *decryption*.

There are a number of methods and levels of encryption. Basic encryption may be broken (or cracked) in a matter of minutes, but some are designed to last much longer—some of the earliest forms of encryption have not been cracked, even after centuries of trying.

If you have banked online, accessed a secure web site that begins with <https://> or submitted a tax return online to the government, you have used data encryption even though you did not encrypt or decrypt the data manually.



Required Reading:

- [Data Encryption](#)
- [Encryption](#) (Read all tabs)
- [History of Cryptography](#)

Definitions: Encryption Key

A data encryption key is a series of text characters used by a mathematical encryption formula to encrypt plain text.

Encryption keys come in 64-bit, 128-bit, and 256-bit. The larger the key, the more difficult it is for someone to crack the code.

Prior to the use of keys, many other forms of encryption (including the cypher book) were used before the availability of computers.

RSA, CRC-32, CRC-16, and MDA are versions of mathematical algorithms used for data encryption.

Required Reading:

- [SSH hardens Secure Shell](#)

Optional Reading:

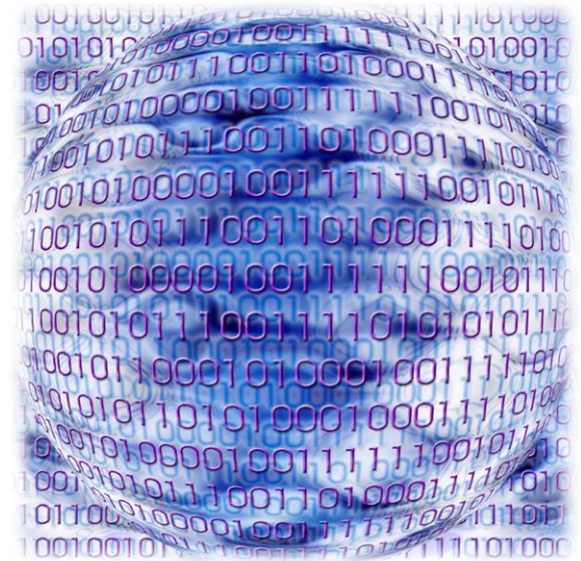
- [RSA encryption](#)
- [RSA wikipedia](#)



Definitions: Algorithm

An algorithm is a formula that ends in a known state or outcome and begins with a known beginning.

One example of an algorithm is a computer program. Programmers write the program with a known set of statements (instructions) that an interpreter follows in order to perform a specific task.



Required Reading:

- [Algorithm](#)
- [Computer algorithm](#)

Definitions: SSH

SSH is a protocol or method of using one computer to access another computer in a secure way across a network. The data transferred between computers is encrypted or coded, which makes it difficult for unauthorized users to access the contents of the transmission.

SSH was designed to be a secure replacement for Telnet, rlogin, rcp, and rsh, which are alternative protocols for network access, but they all have critical risks. SSH provides secure TCP/IP tunnels and is able to compress the data it handles. SSH can also be used with additional authentication schemes to provide highly secure access to any Unix server.

SSH uses a public/private key with RSA authentication to check the identity of the communicating machines and to ensure the encryption of all data exchanged. Additionally a *tunnel* for X11 communications can be setup by SSH by using the proper fields in the SSH config file.



[Ubuntu Documentation](#) > [Community Documentation](#) > [SSH](#)

SSH

Parent page: [Internet and Networking](#)

Introduction

SSH ("Secure SHell") is a protocol for securely accessing one computer from another: command line and graphical programs, transfer files, and even create secure virtual machines.

To use SSH, you will need to install an SSH client on the computer you connect from, and an SSH server on the computer you connect to. The most popular Linux SSH client and Linux SSH server are maintained by OpenSSH.

The OpenSSH client is included in Ubuntu by default. For information on connecting to a remote host, see [OpenSSH Server](#).

To install the OpenSSH server, [install the following package](#): [openssh-server](#).

Required Reading:

- [SSH Part I](#)
- [SSH Part II](#)

Definitions: SSH (Contd)

The proper use of SSH protects against:

- Capturing and stealing data transmitted over the network
- Changing the routers to pretend one machine is another (spoofing)
- Editing and changing data at the routers (provides data integrity)
- Redirecting DNS servers to send traffic to another host instead of the intended one

Unfortunately SSH can not protect against user and setup errors such as:

- Compromised root accounts on servers
- Insecure home directories
- Incorrect configuration



Required Reading:

- [SSH Part I](#)
- [SSH Part II](#)

Definitions: OpenSSH

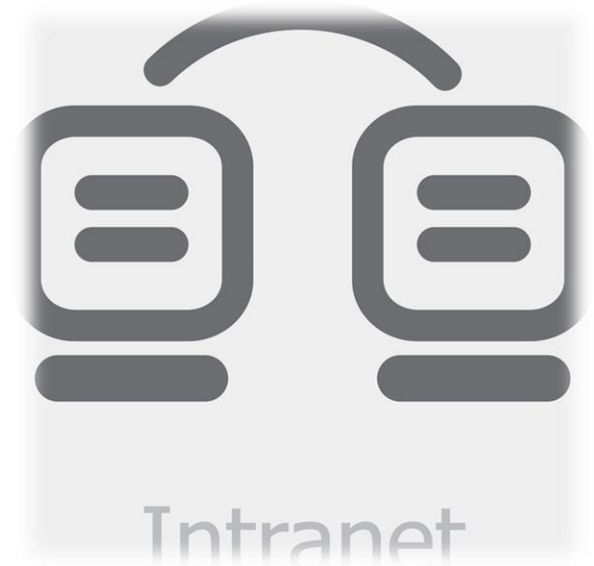
SSH was first implemented as a free protocol called *SSH1*. This version lasted through versions 1.2 and 1.5. In more recent years, the licensing of SSH became very restrictive as the owners, SSH Communications and DataFellows, strongly encouraged users to adapt *SSH2*.

OpenSSH is a free open source alternative to the original SSH and does not have the many restrictions found in the original SSH. OpenSSH is fully compatible and supports both the v1 and v2 protocols and adds new functionality.

Tunneling

The term *tunneling* came from literal underground tunnels or structures that allow people and goods to pass undetected.

SSH tunneling refers to the act of opening an SSH connection from one computer to another and using that tunnel to move data securely from one machine to another. For example, a person who wishes to send a secure email could open a SSH tunnel from machine A to machine B and send the email across that tunnel. The email, though plaintext, would be delivered to machine B without anyone being able to intercept or understand this text.



Required Reading:

- [Open SSH](#)
- [Tunneling with SSH](#)

Definitions: SFTP

FTP is short for *file transfer protocol*. It is a method used by computer users to transfer files across a network from one computer to another. As with any plain text method of transmitting information over the Internet, FTP does not encrypt files. Therefore, files transferred by FTP can be read, stolen, and altered prior to delivery.

Imagine an insurance department transferring a large batch of health information to the State Labor department. If someone was able to intercept this transfer, he/she would have access to confidential diagnosis codes, social security numbers, and addresses, which would result in a serious privacy breach.

SFTP uses a secure, encrypted, SSH connection to transfer a file from machine A to machine B. Because of the encryption and the transmission keys, it becomes difficult for someone to intercept the file(s) being transferred. Additionally, SFTP provides a record of the machine and account that receives its data.



Required Reading:

- [SFTP and its use](#)

Definitions: Telnet

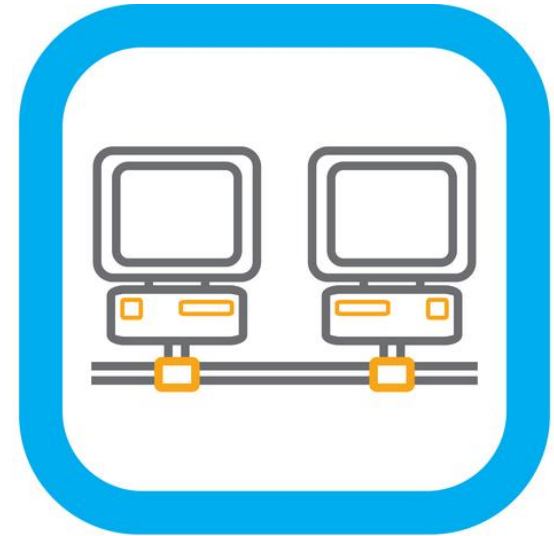
Telnet was one of the original methods of logging into remote terminals. The old telnet command was:

telnet systemname.com

Telnet would connect to the command line of the remote system and allow the user to type in commands, run programs, and develop software using plaintext communications. As the amount of information and the number of people accessing the Internet grew, the sending of plaintext through the Telnet protocol became very insecure because this form of communication could be intercepted.

Consequently, Telnet has been replaced by ssh and other ssh based encrypted terminal programs.

The same problem with clear text (or plaintext) transmissions also applies to rlogin, rcp, rsh, and X11 communication protocols, which are all vulnerable to exposure and interception.



Network

Required Reading:

- [Telnet protocol specification](#)
- [Secure Shell](#)

Definitions: SSH Server

The SSH Server is the machine (or application) to which another connects. It is on the receiving end of the communication. One server can handle multiple SSH connections. The SSH client connects to the server.

The SSH server runs an application named `sshd` that handles all the networking, encryption, and communications.

The client runs either `ssh` or a form of `ssh` (in the case of non-Unix machines) that connects to the server. The client also works with data encryption, keys and machine information to ensure a secure connection.



Required Reading:

- [Telnet protocol specification](#)
- [Secure Shell](#)

Open SSH

Installation Techniques



Installing OpenSSH (Ubuntu)

Many Ubuntu and Debian installations will install OpenSSH by default.

To check your installation, go to the terminal window (**Applications** → **Accessories** → **Terminal**) and type:

Press the [Enter] key after entering each command.

1. **sudo apt-get update**
2. Enter your password when asked
3. **sudo apt-get upgrade**
4. **sudo apt-get install ssh**
5. Enter your password if asked
6. **sudo /etc/init.d/ssh start**

SSH will now be installed and running.



[Ubuntu Documentation](#) > [Community Documentation](#) > [SSHOpenSSHKeys](#)

SSHOpenSSHKeys

Parent page: [Internet and Networking](#) >> [SSH](#)

Public and Private Keys

If your SSH server is visible over the Internet, you should use public key authentication instead of passwords if at all possible. If you don't think that's important, try [logging](#) all of the malicious login attempts you get for the week. My computer - a perfectly ordinary desktop PC - had over 4,000 attempts to guess my password and almost 2,500 break-in attempts in the last week alone. How many thousand random guesses do you think it would take before an attacker stumbles across your password?

With public key authentication, every computer has a public and a private "key" (a large number with particular mathematical properties). The private key is stored on the **.ssh/authorized_keys** file on all the computers. The SSH server uses the public key to "lock" messages in a way that only the most successful attacker can't guess an "authorized" message.

Installing OpenSSH (Fedora)

To install SSH on a Fedora or Redhat machine:

Login to your system and open a terminal window
(**Applications** → **Accessories** → **Terminal**) and type:

Press the [Enter] key after entering each command.

1. **sudo yum upgrade**
2. **sudo yum install ssh**
3. Enter your password when prompted and accept the default options provided by the installation.
4. **sudo /etc/init.d/ssh start**

SSH is now installed and running.

Required Reading:

- [Fedora Install](#)



The screenshot shows the Fedora Wiki page for "Cryptography". The page title is "Cryptography" and it includes a "Contents" table of contents. The page content is partially visible, showing the introduction to cryptographic technology.

fedora™

page discussion view source

WIKI

- [Fedora Project](#)
- [Wiki](#)
- [News](#)
- [Events](#)
- [Features](#)
- [Recent changes](#)
- [Random page](#)
- [Help](#)

NAVIGATION

- [Home](#)
- [Get Fedora](#)
- [Join Fedora](#)

SUB-PROJECTS

- [Ambassadors](#)
- [Bug Zappers](#)

Cryptography

Cryptographic technology helps provide security and privacy with Fedora to provide hard disk encryption.

Contents [hide]

- 1 The Theory
- 2 OpenSSH
 - 2.1 Installing OpenSSH
 - 2.2 Creating SSH Keys
- 3 GnuPG
 - 3.1 Installing GnuPG
 - 3.2 Creating GPG Keys

The Theory

OpenSSH and GnuPG use a common cryptographic...

Available SSH Clients

An SSH client is a terminal application that follows the SSH protocol and can connect to and communicate with a Unix/Linux server running the SSH server application. SSH clients are available for all versions of Microsoft Windows, Linux, and Apple OS/X.

In order to complete the labs in this section you must have a minimum of one running Linux machine (virtual or hardware) and one client. It is possible to complete these labs on a single virtual box machine with two user accounts.

Visit <http://www.openssh.com> for a list of clients and links to download them.

Note: For Apple OS/X, the client is installed at the time of system installation.

- [Appendix A: PuTTY FAQ](#)
 - [A.1 Introduction](#)
 - [A.1.1 What is PuTTY?](#)
 - [A.2 Features supported in PuTTY](#)
 - [A.2.1 Does PuTTY support SSH-2?](#)
 - [A.2.2 Does PuTTY support reading OpenSSH or ssh.c](#)
 - [A.2.3 Does PuTTY support SSH-1?](#)
 - [A.2.4 Does PuTTY support local echo?](#)
 - [A.2.5 Does PuTTY support storing settings, so I don't ha](#)
 - [A.2.6 Does PuTTY support storing its settings in a disk f](#)
 - [A.2.7 Does PuTTY support full-screen mode, like a DO](#)
 - [A.2.8 Does PuTTY have the ability to remember my pas](#)
 - [A.2.9 Is there an option to turn off the annoying host key](#)
 - [A.2.10 Will you write an SSH server for the PuTTY suite](#)
 - [A.2.11 Can PSCP or PSFTP transfer files in ASCII mo](#)

Required Reading:

- [PuTTY](#)

Configuring the SSH Server

In order to setup a secure SSH environment you should edit the default SSH configuration file that was installed when the SSH package was installed. The configuration is the same regardless of the Linux distribution or the version of client you are running.

In order to begin your SSH configuration, log into the server machine through a terminal window. Then type:

```
cd /etc/ [enter]
```

All of our configuration steps will begin at this point.

- If you are on an Ubuntu server you will find all of your configuration files in `/etc/ssh/`
- If you are on a Fedora server you will find all of your configuration files in `/etc/ssh/`
- If you are on an OS/X machine you will find your configuration files in `/etc/`



[Ubuntu Documentation](#) > [Ubuntu 8.04 LTS](#) > [Ubuntu Server Guide](#) > [Remote Adminis](#)

OpenSSH Server

Introduction

This section of the Ubuntu Server Guide introduces a powerful collection of tools for t computers and transfer of data between networked computers, called *OpenSSH*. You configuration settings possible with the OpenSSH server application and how to chan

OpenSSH is a freely available version of the Secure Shell (SSH) protocol family of tc transferring files between computers. Traditional tools used to accomplish these func insecure and transmit the user's password in cleartext when used. OpenSSH provide: facilitate secure, encrypted remote control and file transfer operations, effectively rep

The OpenSSH server component, **sshd**, listens continuously for client connections fr connection request occurs, **sshd** sets up the correct connection depending on the ty example, if the remote computer is connecting with the **ssh** client application, the Ope session after authentication. If a remote user connects to an OpenSSH server with **sc** a secure copy of files between the server and client after authentication. OpenSSH c: including plain password, public key, and **Kerberos** tickets.

Installation

Installation of the OpenSSH client and server applications is simple. To install the Op system, use this command at a terminal prompt:

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

Required Reading:

- [SShd-config manual page](#)
- [SSH Configuration](#)
- [Configure SSH for Beginners](#)

SSHD Configuration File

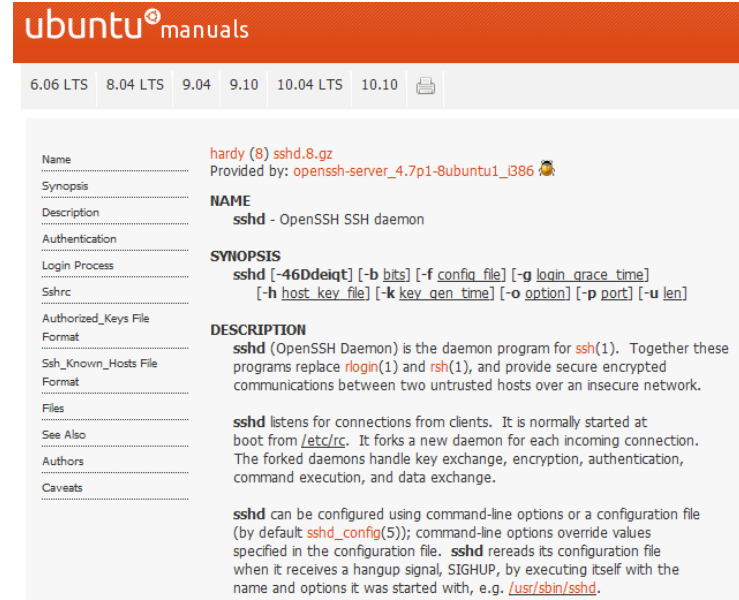
The `sshd_config` file is either found in `/etc/` or `/etc/ssh` and is the system-wide configuration file for OpenSSH. Modifying this file allows you to modify the operation of the ssh daemon (server).

The file is made up of keywords and value—one keyword and one value per line. When you examine the file, the keywords will be both uppercase and lowercase, but this is for ease of reading. In reality, the keywords are not case sensitive.

An example of these lines is:

```
Port 22
ListenAddress 192.168.1.1
HostKey /etc/ssh/ssh_host_key
ServerKeyBits 1024
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
IgnoreRhosts yes
IgnoreUserKnownHosts yes
StrictModes yes
```

← Note, the keyword followed by a space, followed by the value.



The screenshot shows the Ubuntu manual page for the `sshd` daemon. The page title is "ubuntu manuals" and it lists various Ubuntu versions (6.06 LTS, 8.04 LTS, 9.04, 9.10, 10.04 LTS, 10.10). The manual page content includes:

- Name:** `sshd` (8) `sshd.8.gz`
- Provided by:** `openssh-server_4.7p1-8ubuntu1_0386`
- NAME:** `sshd` - OpenSSH SSH daemon
- SYNOPSIS:** `sshd [-46Ddeiq] [-b bits] [-f config_file] [-g login_grace_time] [-h host_key_file] [-k key_gen_time] [-o option] [-p port] [-u len]`
- DESCRIPTION:** `sshd` (OpenSSH Daemon) is the daemon program for `ssh(1)`. Together these programs replace `rlogin(1)` and `rsh(1)`, and provide secure encrypted communications between two untrusted hosts over an insecure network. `sshd` listens for connections from clients. It is normally started at boot from `/etc/rc`. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange. `sshd` can be configured using command-line options or a configuration file (by default `sshd_config(5)`); command-line options override values specified in the configuration file. `sshd` rereads its configuration file when it receives a hangup signal, `SIGHUP`, by executing itself with the name and options it was started with, e.g. `/usr/sbin/sshd`.

Required Reading:

- [SShd-config manual page](#)

Editing the Config File

To edit the SSHD configuration file:

Open a terminal and type:

```
cd /etc/ [enter]  
ls sshd_config [enter]
```

If the file name is returned, then the file is in */etc/*

If you get *no files found* or *file not found*, type:

```
ls /etc/ssh/sshd_config [enter]
```

If the file is not found, you will need to install SSH.

Remember the location of your *sshd_config* file.

Select **PLAY** below to view a video on SSHD configuration.



View Video
VideoLesson7SSHDConfiguration(C2L7S32).mp4

Required Reading:

- [SSD manual page](#)

Important Terms

Find and examine the following lines:

Port – the port number that sshd listens on. The default is 22. Leave it at this value unless you have a reason to assign another number. Do not assign it a port number under 600 since ports below 600 may be taken by another application.

ListenAddress – this is the address of the interface on which sshd will listen. For example, if the machine has two network interface cards, both addresses may be listed here. If the machine has a network interface card with an internal ip address (local network) and a network interface card with an external ip address (wide area public network), the administrator may choose to put the ip address of the local network here. This action would prevent others from accessing the machine from the public network.

NOTE: Be careful not to assign a value of 127.0.0.1 or you will not be able to log into your machine without having access to the console.

Configuration

You may configure the default behavior of the OpenSSH server application, **sshd**, by editing the configuration file `/etc/ssh/sshd_config`. For information about the configuration directives in this 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 authentication modes. The following are examples of configuration directives in the `/etc/ssh/sshd_config` file.



Prior to editing the configuration file, you should make a copy of the file so you will have the original settings as a reference and to avoid losing any changes.

Copy the `/etc/ssh/sshd_config` file and protect it from writing with the following command, 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
```

Important Terms (Contd)

Additional options in the SSHD config file:

ServerKeyBits – this is the number of bits that the server uses to generate the encryption key when it starts. The higher the better. Don't exceed 2048.

LoginGraceTime – how long in seconds after a connection that the server will wait before disconnecting if the user does not successfully log in.

PermitRootLogin – This option determines whether the root user can log in through ssh or not. Do not ever set this option to “yes” because allowing root logins is extremely insecure.

X11Forwarding – Set to “yes” if you wish to allow X11 forwarding which allows remote users to run GUI's on the remote server without installing it on their own machines.

RSAAuthentication – Set to “yes” at all times. This forces RSA authentication for strong security.

PasswordAuthentication – Set to “yes” at all times to enforce password authentication for stronger security.

AllowUsers – If you want to restrict the users that can access the host through ssh, enter that information here. Multiple users are separated by spaces.

Creating Public and Private Keys

The next step in configuring SSH for secure connections to a remote server is to generate a public and private key for your user id.

1. Open a terminal window on your on your local (client) machine.
2. Type **ssh-keygen -t rsa** [enter]
3. Accept the default value for *file name* and *location*, unless you have a reason to make a change.
4. Enter a strong passphrase. For greater security, enter a phrase that is not guessable and has more than one word. Make sure you remember what you entered. You may also leave the passphrase prompt blank if you do not wish.
5. Copy the newly created public key to the `.ssh` directory on the server. In the terminal window, type:
scp ~/.ssh/id_rsa.pub user@machine.com:~/.ssh/authorized_keys
6. Test by typing: **ssh user@machine.com**
7. The SCP command is a secure, encrypted copy command that is ssh based. Make sure you change user to your user id on the remote machine



Select **PLAY** below to view a video on encryption keys.

View Video
VideoLesson7EncryptionKeys(C2L7S35).mp4

You should now be able to connect to the remote machine once you type your passphrase (if you used one). If you cannot connect, there are a few different troubleshooting steps you should explore.

Troubleshooting SSH Server

If for some reason you are unable to connect once you copy your public and private keys to the server, there are a few steps you should take.

1. Open the terminal window.
2. Type `ssh user@machine.com` to connect to the server.
3. Change the `user@machine.com` to the actual user ID and machine that has your account.
4. Verify that the `ssh_config` file is setup properly and that you have restarted `ssh` after making any changes.

Most problems that arise during configuration occurs because of the choices made in the configuration files.

Once you are able to log into the server using your `ssh` key file, you can continue to use other `ssh`-based utilities.

Configuration

You may configure the default behavior of the OpenSSH server application, `sshd`, by editing the configuration file `/etc/ssh/sshd_config`. For information about the configuration directives, see the 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 authentication modes. The following are examples of configuration directives that you can use in the `/etc/ssh/sshd_config` file.



Prior to editing the configuration file, you should make a copy of the file so you will have the original settings as a reference and to avoid losing your changes.

Copy the `/etc/ssh/sshd_config` file and protect it from writing by running the following command at a terminal prompt:

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original
sudo chmod a-w /etc/ssh/sshd_config.original
```

SSH Tunnel

A ssh tunnel allows you to securely access remote content through an ssh connection on a remote host. An SSH tunnel is used to access a web page on a secure intranet server within a company's firewall. An SSH tunnel might also be used to access an internal shared partition on a company's internal network.

We begin by setting up the ssh tunnel itself.

Type:

```
ssh -L 2022:machine1.com:80 authuser@machine.com
```

This command directs any traffic hitting port 2022 on the local machine to go to port 80 on *machine1.com*. The user information is also contained in the command. Port 80 is used by the web server.



Select **PLAY** below to view a video on tunneling.

View Video
VideoLesson7SSHTunnel(C2L7S
37).mp4

Required Reading:

- [SSH tutorial](#)

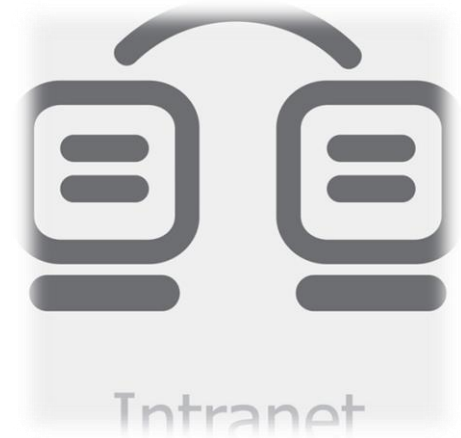
Configuring SSH Tunnel

There are times where you or your users will want to use X11 GUI's from remote locations. SSH can be configured to encrypt and provide this connection.

To setup this connection, make sure that you have X11Forwarding set to yes in the `sshd_config` file. Then:

1. Open the terminal window from xwindows on your local (client) machine.
2. Type **`ssh -X suser@remotemachine.com`** [enter]
3. Type the name of the command that you would like to run from the command line on the remote machine . An example of this may be **`system-config-kickstart`** [enter].

This is one way to run GUIs while not having to install and reprogram them on the local machine. For example, a user with a Windows machine could run a Linux application on the remote machine. A second user on an OS/X machine could run the same application on the server across an SSH connection at the same time.



Required Reading:

- [SSH tutorial](#)
- [Beginning SSH on Ubuntu](#)

Lesson Summary

In this lesson, you explored the need for encryption and the different encryption methods available for computer users. The main reason for most encryption is to prevent plaintext transmissions of data over the Internet to protect the content and the systems carrying it. Data theft is a big business and can be used for corporate espionage, identity theft, and information leaks.

The most effective methods of data security for terminal sessions, X window connections, and web transmission is to use SSH, a secure transfer/transmission utility.

You also learned to configure the `sshd_config` file, generate your public and private keys, put these keys on the server, use `ssh` to connect to the server, use tunneling features, and forward X11 connections and applications.

