Jackson State University
Department of Computer Science
CSC 437-01/539-01 Computer Security
Fall 2013
Instructor: Dr. Natarajan Meghanathan

Lab Project # 2: Running Secure Shell (SSH) Server in a Virtual Machine Environment

Due: October 22, 2013: 7.30 PM

This project is for educational and awareness purposes only. We are not responsible for anyone using this project for any malicious intent. The objective of this project is to educate students how to install and run SSH in a Linux virtual machine environment as well as remotely connect from a Windows machine. This project description encloses two tutorial documents that you can use as reference to install and run the SSH commands. You will need to download VMware Player which is the virtualization software that will be used for this project. You will also need to download CentOS and Ubuntu OS to complete this project. We will use Windows 7 as the host operating system.

Installing VMWare Player

Download the latest version (v.5 or v.6) of VMware Player for your Operating System from https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/5_0

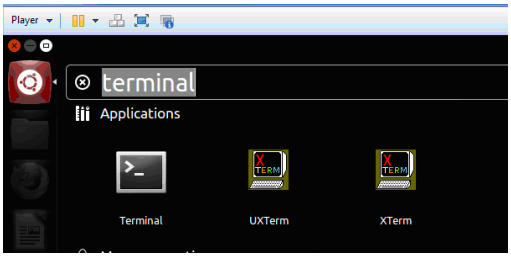
Downloading and Installing Ubuntu OS

1. Download Ubuntu OS http://www.ubuntu.com/download/desktop and save it somewhere on your computer
2. Open up VMWare Player
3. Click on **Create a New Virtual Machine**
4. Select Installer disc image file (iso): browse for your Ubuntu .iso file and click **Next**
5. Type in your full name in the space provided. Use your J-number as Username (with a lowercase j). In my case, I use **natarajan** as the username. For your password, Select a password of your choice (easy to remember; but, difficult to find out by others). Click **Next** after entering the information.
6. Next, type in a name for your virtual machine (use your J-number again). Click **Next**.
7. On the next page, select **Store virtual disk as a single file**, and click **Next**.
8. Click **Finish** on the next page and wait for the OS to be installed.
9. Next, log into Ubuntu OS with your password and press **Enter**.

10. Click the **Player** menu, and go to **Manage** then **Virtual Machine settings.**

11. When the settings come up, make sure that the **Network Adapter** is set to **NAT**, and click **OK**.

12. Launch a terminal by clicking the **Dash Home** (indicated in the picture below) and typing **terminal** in the box provided. Then click the **Terminal** icon.



Downloading and Installing CentOS

1. Download CentOS (CentOS-6.4-i386-LiveCD.iso) http://centos.icyboards.com/6.4/isos/i386/ and save it somewhere on your computer
2. Open up VMWare Player
3. Click on **Create a New Virtual Machine**
4. Select Installer disc image file (iso): browse for your CentOS .iso file and click **Next**
5. On the next page, select **Store virtual disk as a single file**, and click **Next**.

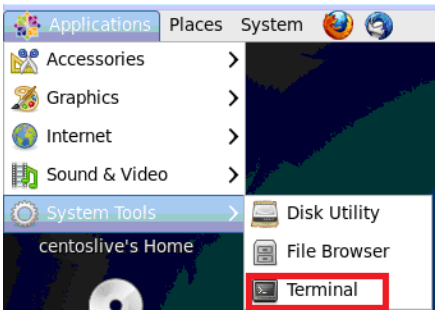
6. Click **Finish** on the next page and wait for the OS to be installed.

7. You can setup automatic login without requiring a password. If you wish to setup a password, you could also do so. You should be now logged into the CentOS system.

8. Click the **Player** menu, and go to **Manage** then **Virtual Machine settings.**

9. When the settings come up, make sure that the **Network Adapter** is set to **NAT**, and click **OK**.

10. Launch a terminal from the Applications --> System --> Terminal menu.



Installing OpenSSH Server in the Ubuntu VM

1) You need to setup root access in the Ubuntu VM. To setup root access, run the command **su passwd root** on the terminal. Enter the password you setup to login to the Ubuntu VM as a regular user (in my case, the username of the regular user is natarajan). Then, setup a password for the **root** level access and confirm it. The screenshot is shown below.

```
natarajan@ubuntu:~$ sudo passwd root
[sudo] password for natarajan:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
natarajan@ubuntu:~$
```

2) Login as root using the command **su root**.

```
natarajan@ubuntu:~$ su root
Password:
root@ubuntu:/home/natarajan#
```

Tasks to do:

From now on, you would refer to the two tutorial documents and any other needed reference posted in the Internet. You need to do the tasks in the order specified below (increasing order of task numbers). Wherever hostname is needed to run the SSH/SCP commands, you could use the IP address of the corresponding host.

Note: Make sure the welcome message that you configure in Task 3 (the welcome message should appear when someone requests for a SSH connection to the Ubuntu VM) appears when you attempt to remotely connect to and/or remotely print the contents of a file in the Ubuntu VM using SSH or SCP when you execute Tasks 8, 11, 12 and 13.

Task 1: Install the OpenSSH server application on the Ubuntu VM. After the installation is complete, run the **netstat –ntlp** command to show that the SSH daemon (sshd) is one of the programs actively running on a tcp port listening for incoming connection requests. Identify the port number on which the **sshd** daemon is running. Include appropriate screenshot(s).

Task 2: Find out the IP address of the Ubuntu VM by running the **ifconfig** command. Include a screenshot showing the IP address.

Task 3: Configure the OpenSSH Server application to greet a user connecting to the server with a customized welcome message: **Welcome to the Ubuntu Virtual Machine**. Explain in detail the sequence of commands you ran and the changes to the files that you made to accomplish this.

Task 4: After the changes made to the SSH configuration (as required for Task 3), start the **sshd** server application to effect the changes as well as for remote machines to connect. Include screenshot.

Task 5: Exit out of the root privileges in the Ubuntu VM. Create a text file that has the name ubuntu_1234.txt where in your case, 1234 should be replaced by the last four digits of your J#. Open the text file in an editor and type a sentence that includes your name: "This is a text file in the Ubuntu VM – created by <Your Full Name>". Save and exit the text editor. Run the **cat** command locally on the Ubuntu VM to print the contents of this text file on the terminal. Include screenshot.

Task 6: Launch a terminal in the CentOS VM. Find the IP address of the VM. Include a screenshot.

Task 7: Now, create a text file (similar to Task 5) in the CentOS VM under the default working directory of the regular user, with a file name like centos_1234.txt where 1234 should correspond in your case to the last 4 digits of your J#. Run the **cat** command locally on the CentOS VM to print the contents of this text file on the terminal. Include screenshot.

Task 8: On the CentOS VM, run the appropriate **ssh** command to remotely connect/login to the Ubuntu VM as a regular user (in my case, natarajan) and take appropriate screenshots, including one to show the modified welcome message. Make sure to type 'yes' when queried during the connection attempt.

Task 9: After you logged into the Ubuntu VM (as per Task 7) from the CentOS VM, run a **cat** command to print the contents of the ubuntu_1234.txt file. Include screenshot.

Task 10: Log out (exit) of the Ubuntu VM SSH connection and return to the CentOS terminal as a regular user. Include screenshot.

Task 11: From the Cent OS VM terminal, run an appropriate **scp** command to transfer the centos_1234.txt file from the default/current working directory of the regular user in the CentOS VM to the present working directory of the regular user in the Ubuntu VM, under a new file name like centos_ubuntu_1234.txt file. Include screenshot.

Task 12: From the Cent OS VM terminal, run an appropriate **scp** command to transfer the ubuntu_1234.txt file from the Ubuntu VM to the default/current working directory of the CentOS VM, under a new file name like ubuntu_centos_1234.txt file. Print the contents of the ubuntu_centos_1234.txt file in the CentOS VM. Include screenshots for both the file transfer and print command.

Task 13: Without logging in to the Ubuntu VM, run the appropriate SSH command from the Cent OS VM itself to remotely print (cat) the contents of the two text files in the Ubuntu VM: ubuntu_1234.txt and centos_ubuntu_1234.txt file. Include screenshots.

Task 14: On your host Windows machine, launch the SSH Secure Shell client application to connect to the Ubuntu VM as a regular user (in my case, natarajan) and transfer the above two text files: ubuntu_1234.txt and centos_ubuntu_1234.txt file to the Windows machine. Show screenshot of logging in to the Ubuntu VM from SSH Client application in Windows.

Task 15: Print the contents of the two text files (that you transferred in Task 14) in a DOS prompt using the **type** command. Include screenshot.

What to Submit: Your answers to the questions and responses for the activities to be conducted in each of the tasks 1 through 15. Include appropriate screenshots.

Search

Official Documentation

OpenSSH Server

Introduction

This section of the Ubuntu Server Guide introduces a powerful collection of tools for the remote control of networked computers 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 a computer 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 client applications on your Ubuntu system, use this command at a terminal prompt:

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

To install the OpenSSH server application, and related support files, use this command at a terminal prompt:

```
sudo apt-get install openssh-server
```

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

  In the `/etc/ssh/sshd_config` file, or if already present, ensure the line 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 /etc/init.d/ssh restart
```

Many other configuration directives for **sshd** are available for changing 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, or that the **sshd** server refuses to start due to an incorrect configuration directive, 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 dsa
```

This will generate the keys using a *DSA* authentication identity of the user. 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_dsa.pub`, while `~/.ssh/id_dsa` is the *private* key. Now copy the `id_dsa.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.

References

- Ubuntu Wiki SSH page.
- OpenSSH Website
- Advanced OpenSSH Wiki Page

Chapter 5. Remote Administration eBox

21.3. Configuring an OpenSSH Client

To connect to an OpenSSH server from a client machine, you must have the `openssh-clients` and `openssh` packages installed on the client machine.

21.3.1. Using the `ssh` Command

The `ssh` command is a secure replacement for the `rlogin`, `rsh`, and `telnet` commands. It allows you to log in to a remote machine as well as execute commands on a remote machine.

Logging in to a remote machine with `ssh` is similar to using `telnet`. To log in to a remote machine named penguin.example.net, type the following command at a shell prompt:

```
ssh penguin.example.net
```

The first time you `ssh` to a remote machine, you will see a message similar to the following:

```
The authenticity of host 'penguin.example.net' can't be established.
DSA key fingerprint is 94:68:3a:3a:bc:f3:9a:9b:01:5d:b3:07:38:e2:11:0c.
Are you sure you want to continue connecting (yes/no)?
```

Type **`yes`** to continue. This will add the server to your list of known hosts (`~/.ssh/known_hosts/`) as seen in the following message:

```
Warning: Permanently added 'penguin.example.net' (RSA) to the list of known hosts.
```

Next, you will see a prompt asking for your password for the remote machine. After entering your password, you will be at a shell prompt for the remote machine. If you do not specify a username the username that you are logged in as on the local client machine is passed to the remote machine. If you want to specify a different username, use the following command:

```
ssh username@penguin.example.net
```

You can also use the syntax `ssh -l username penguin.example.net`.

The `ssh` command can be used to execute a command on the remote machine without logging in to a shell prompt. The syntax is `ssh hostname command`. For example, if you want to execute the command `ls /usr/share/doc` on the remote machine penguin.example.net, type the following command at a shell prompt:

```
ssh penguin.example.net ls /usr/share/doc
```

After you enter the correct password, the contents of the remote directory `/usr/share/doc` will be displayed, and you will return to your local shell prompt.

21.3.2. Using the `scp` Command

The `scp` command can be used to transfer files between machines over a secure, encrypted connection. It is similar to `rcp`.

The general syntax to transfer a local file to a remote system is as follows:

```
scp <localfile> username@tohostname:<remotefile>
```

The *<localfile>* specifies the source including path to the file, such as `/var/log/maillog`. The *<remotefile>* specifies the destination, which can be a new filename such as `/tmp/hostname-maillog`. For the remote system, if

you do not have a preceding `/`, the path will be relative to the home directory of *username*, typically `/home/username/`.

To transfer the local file `shadowman` to the home directory of your account on penguin.example.net, type the following at a shell prompt (replace *username* with your username):

```
scp shadowman username@penguin.example.net:shadowman
```

This will transfer the local file `shadowman` to `/home/username/shadowman` on penguin.example.net. Alternately, you can leave off the final `shadowman` in the `scp` command.

The general syntax to transfer a remote file to the local system is as follows:

```
scp username@tohostname:<remotefile> <newlocalfile>
```

The *<remotefile>* specifies the source including path, and *<newlocalfile>* specifies the destination including path.

Multiple files can be specified as the source files. For example, to transfer the contents of the directory `downloads/` to an existing directory called `uploads/` on the remote machine penguin.example.net, type the following at a shell prompt:

```
scp downloads/* username@penguin.example.net:uploads/
```

21.3.3. Using the `sftp` Command

The `sftp` utility can be used to open a secure, interactive FTP session. It is similar to `ftp` except that it uses a secure, encrypted connection. The general syntax is *`sftp username@hostname.com`*. Once authenticated, you can use a set of commands similar to those used by FTP. Refer to the `sftp` man page for a list of these commands. To read the man page, execute the command `man sftp` at a shell prompt. The `sftp` utility is only available in OpenSSH version 2.5.0p1 and higher.

21.3.4. Generating Key Pairs

If you do not want to enter your password every time you use `ssh`, `scp`, or `sftp` to connect to a remote machine, you can generate an authorization key pair.

Keys must be generated for each user. To generate keys for a user, use the following steps as the user who wants to connect to remote machines. If you complete the steps as root, only root will be able to use the keys.

Starting with OpenSSH version 3.0, `~/.ssh/authorized_keys2`, `~/.ssh/known_hosts2`, and `/etc/ssh_known_hosts2` are obsolete. SSH Protocol 1 and 2 share the `~/.ssh/authorized_keys`, `~/.ssh/known_hosts`, and `/etc/ssh/ssh_known_hosts` files.

Red Hat Enterprise Linux 4 uses SSH Protocol 2 and RSA keys by default.

Tip

If you reinstall and want to save your generated key pair, backup the `.ssh` directory in your home directory. After reinstalling, copy this directory back to your home directory. This process can be done for all users on your system, including root.

21.3.4.1. Generating an RSA Key Pair for Version 2

Use the following steps to generate an RSA key pair for version 2 of the SSH protocol. This is the default starting with OpenSSH 2.9.

1. To generate an RSA key pair to work with version 2 of the protocol, type the following command at a shell prompt:

```
ssh-keygen -t rsa
```

Accept the default file location of `~/.ssh/id_rsa`. Enter a passphrase different from your account password and confirm it by entering it again.

The public key is written to `~/.ssh/id_rsa.pub`. The private key is written to `~/.ssh/id_rsa`. Never distribute your private key to anyone.

2. Change the permissions of the `.ssh` directory using the following command:

```
chmod 755 ~/.ssh
```

3. Copy the contents of `~/.ssh/id_rsa.pub` into the file `~/.ssh/authorized_keys` on the machine to which you want to connect. If the file `~/.ssh/authorized_keys` exist, append the contents of the file `~/.ssh/id_rsa.pub` to the file `~/.ssh/authorized_keys` on the other machine.

4. Change the permissions of the `authorized_keys` file using the following command:

```
chmod 644 ~/.ssh/authorized_keys
```

5. If you are running GNOME, skip to Section 21.3.4.4 *Configuring `ssh-agent` with GNOME*. If you are not running the X Window System, skip to Section 21.3.4.5 *Configuring `ssh-agent`*.

21.3.4.2. Generating a DSA Key Pair for Version 2

Use the following steps to generate a DSA key pair for version 2 of the SSH Protocol.

1. To generate a DSA key pair to work with version 2 of the protocol, type the following command at a shell prompt:

```
ssh-keygen -t dsa
```

Accept the default file location of `~/.ssh/id_dsa`. Enter a passphrase different from your account password and confirm it by entering it again.

Tip

A passphrase is a string of words and characters used to authenticate a user. Passphrases differ from passwords in that you can use spaces or tabs in the passphrase. Passphrases are generally longer than passwords because they are usually phrases instead of a single word.

The public key is written to `~/.ssh/id_dsa.pub`. The private key is written to `~/.ssh/id_dsa`. It is important never to give anyone the private key.

2. Change the permissions of the `.ssh` directory with the following command:

```
chmod 755 ~/.ssh
```

3. Copy the contents of `~/.ssh/id_dsa.pub` into the file `~/.ssh/authorized_keys` on the machine to which you want to connect. If the file `~/.ssh/authorized_keys` exist, append the contents of the file `~/.ssh/id_dsa.pub` to the file `~/.ssh/authorized_keys` on the other machine.

4. Change the permissions of the `authorized_keys` file using the following command:

```
chmod 644 ~/.ssh/authorized_keys
```

5. If you are running GNOME, skip to Section 21.3.4.4 *Configuring `ssh-agent` with GNOME*. If you are not running the X Window System, skip to Section 21.3.4.5 *Configuring `ssh-agent`*.

21.3.4.3. Generating an RSA Key Pair for Version 1.3 and 1.5

Use the following steps to generate an RSA key pair, which is used by version 1 of the SSH Protocol. If you are only connecting between systems that use DSA, you do not need an RSA version 1.3 or RSA version 1.5 key pair.

1. To generate an RSA (for version 1.3 and 1.5 protocol) key pair, type the following command at a shell prompt:

   ```
   ssh-keygen -t rsa1
   ```

   Accept the default file location (`~/.ssh/identity`). Enter a passphrase different from your account password. Confirm the passphrase by entering it again.

   The public key is written to `~/.ssh/identity.pub`. The private key is written to `~/.ssh/identity`. Do not give anyone the private key.

2. Change the permissions of your `.ssh` directory and your key with the commands `chmod 755 ~/.ssh` and `chmod 644 ~/.ssh/identity.pub`.

3. Copy the contents of `~/.ssh/identity.pub` into the file `~/.ssh/authorized_keys` on the machine to which you wish to connect. If the file `~/.ssh/authorized_keys` does not exist, you can copy the file `~/.ssh/identity.pub` to the file `~/.ssh/authorized_keys` on the remote machine.

4. If you are running GNOME, skip to Section 21.3.4.4 *Configuring `ssh-agent` with GNOME*. If you are not running GNOME, skip to Section 21.3.4.5 *Configuring `ssh-agent`*.

21.3.4.4. Configuring `ssh-agent` with GNOME

The `ssh-agent` utility can be used to save your passphrase so that you do not have to enter it each time you initiate an `ssh` or `scp` connection. If you are using GNOME, the `openssh-askpass-gnome` package contains the application used to prompt you for your passphrase when you log in to GNOME and save it until you log out of GNOME. You will not have to enter your password or passphrase for any `ssh` or `scp` connection made during that GNOME session. If you are not using GNOME, refer to Section 21.3.4.5 *Configuring `ssh-agent`*.

To save your passphrase during your GNOME session, follow the following steps:

1. You will need to have the package `openssh-askpass-gnome` installed; you can use the command `rpm -q openssh-askpass-gnome` to determine if it is installed or not. If it is not installed, install it from your Red Hat Enterprise Linux CD-ROM set, from a Red Hat FTP mirror site, or using Red Hat Network.

2. Select **Main Menu Button** (on the Panel) => **Preferences** => **More Preferences** => **Sessions**, and click on the **Startup Programs** tab. Click **Add** and enter **`/usr/bin/ssh-add`** in the **Startup Command** text area. Set it a priority to a number higher than any existing commands to ensure that it is executed last. A good priority number for `ssh-add` is 70 or higher. The higher the priority number, the lower the priority. If you have other programs listed, this one should have the lowest priority. Click **Close** to exit the program.

3. Log out and then log back into GNOME; in other words, restart X. After GNOME is started, a dialog box will appear prompting you for your passphrase(s). Enter the passphrase requested. If you have both DSA and RSA key pairs configured, you will be prompted for both. From this point on, you should not be prompted for a password by `ssh`, `scp`, or `sftp`.

21.3.4.5. Configuring `ssh-agent`

The `ssh-agent` can be used to store your passphrase so that you do not have to enter it each time you make a `ssh` or `scp` connection. If you are not running the X Window System, follow these steps from a shell prompt. If you are running GNOME but you do not want to configure it to prompt you for your passphrase when you log in (refer to Section 21.3.4.4 *Configuring `ssh-agent` with GNOME*), this procedure will work in a terminal window, such as an XTerm. If you are running X but not GNOME, this procedure will work in a terminal window. However, your passphrase will only be remembered for that terminal window; it is not a global setting.

1. At a shell prompt, type the following command:

   ```
   exec /usr/bin/ssh-agent $SHELL
   ```

2. Then type the command:

```
ssh-add
```

and enter your passphrase(s). If you have more than one key pair configured, you will be prompted for each one.

3. When you log out, your passphrase(s) will be forgotten. You must execute these two commands each time you log in to a virtual console or open a terminal window.

Prev Configuring an OpenSSH Server | Home Up | Next Additional Resources