# FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

**Department of Computer Engineering**

# Course, Subject & Experiment Details

|  |  |
| --- | --- |
| **Practical No:** | **8** |
| **Title:** | **Configuring SSH servers on Linux Machine** |
| **Name of the Student:** | **Warren Fernandes** |
| **Roll No:** | **8940** |
| **Date of Performance:** | **31-01-2022** |
| **Date of Submission:** | **11-04-2022** |

Evaluation:

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Rubric** | **Grade** |
| **1** | **On time submission/completion (2)** |  |
| **2** | **Preparedness (2)** |  |
| **3** | **Skill (4)** |  |
| **4** | **Output (2)** |  |

# Signature of the Teacher

# What is SSH?

* SSH stands for Secure Shell.
* SSH is a network protocol for secure data communication.
* SSH protocol allows remote command-line login.
* SSH protocol enables remote command execution.
* To use SSH you need to deploy SSH Server and SSH Client program respectively.
* OpenSSH is a FREE version of the SSH.
* Telnet, rlogin, and FTP transmit unencrypted data over the internet.
* OpenSSH encrypts data before sending it over insecure network like the internet.
* OpenSSH effectively eliminates eavesdropping, connection hijacking, and other attacks.
* OpenSSH provides secure tunnelling and several authentication methods.
* OpenSSH replaces Telnet and rlogin with SSH, RCP with SCP, FTP with sftp.

# Terminologies used

## 

## sshd: The daemon service that implements the ssh server. By default it must be listening on port 22 TCP/IP.

## ssh: The ssh [ Secure Shell command ] is a secure way to log and execute commands in to SSH Server system.

## scp: The Secure Copy command is a secure way to transfer files between computers using the private/public key encryption method.

## ssh-keygen: This utility is used to create the public/private keys.

## 

## ssh-agent: This utility holds private keys used for RSA authentication.

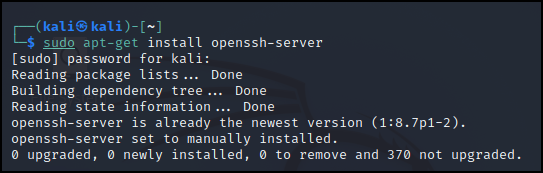
## ssh-add: Adds RSA identities to the authentication agent ssh-agent.

# Requirements:

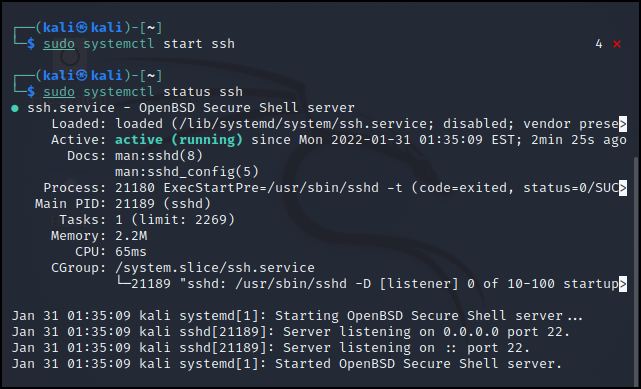
* + Two Linux Virtual Machines installed on Virtual-Box or VMware
  + Openssh-server installed on both the VMs (if not then follow next section)
  + To have both the VMs have different IP address make sure to enable NAT Network settings in VirtualBox.

# How to configure SSH client on Linux?

Install openssh-server using apt-get



Check the current status of **sshd** service, it must be running. If service is stopped start it. Options you need with service command are **start** | **stop** | **restart** | **status**

****

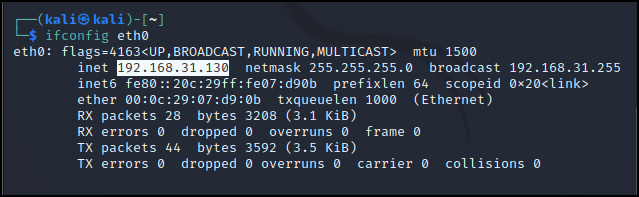
\*NOTE: if above command show error then use –

“sudo systemctl start ssh” OR “sudo systemctl status ssh” (without double quotes)

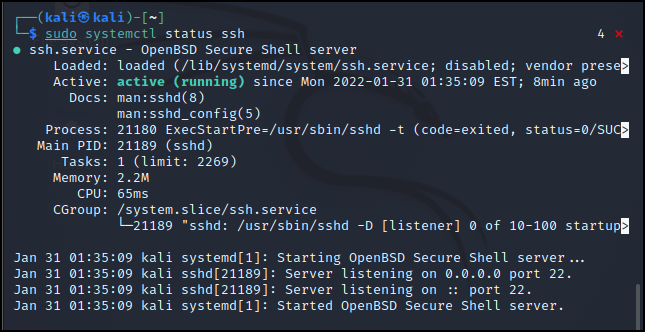
Configure it to start when the system is booted - **sudo systemctl enable ssh**

chkconfig-sshd-server

IP address of OpenSSH server is required, note it down



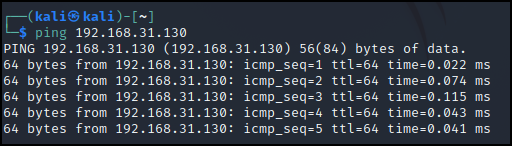
Check **sshd** service status it must be running. Start it if it is off



Configure **sshd** service to start to at boot time (optional)

service-sshd-status-client

Check connectivity from SSH server



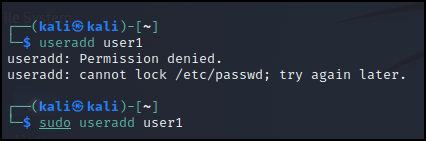
That's all setting which we need on client system.

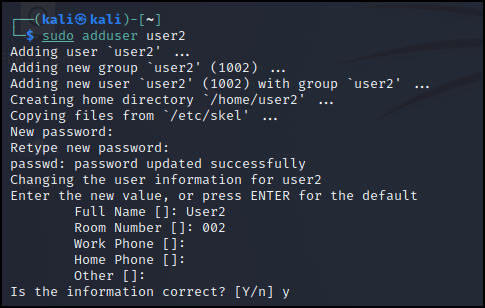
Create two user user1 and user2 and verify that both users can login in SSH server from SSH client.

Go on server and create two users **user1** and **user2 (Use sudo -otherwise it throws error)**

**Adduser – interactive->home directory will be created**

**Useradd – not interative ->home directory will not be created**





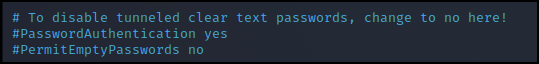
Open main configuration file **sshd\_config**

****

vi-sshd-config

\*NOTE: ‘vi’ / ‘vim’ is a CMD line editor.

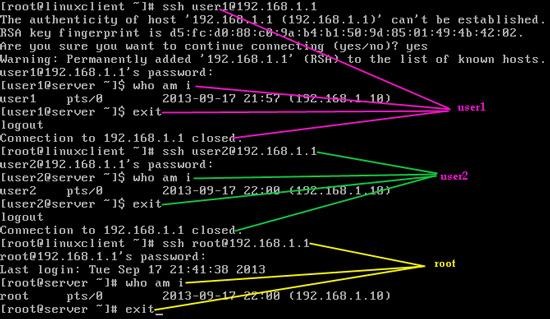
Check the value of **PasswordAuthentication** directive. In order to accept local user password base authentication, it must be set to **yes**. Set it to **yes** if it is set to **no** and save the file.



Restart the service if you have made any change in **sshd\_config**

****

Go on **linuxclient** system and verify that both users can login in SSH server. Also verify from **root** user.



Do not allow root and user1 users to login to it and allow the rest of users. To confirm it login from user2.

## User and Host Based Security

Following additional directives can be added to **/etc/sshd/sshd\_config** file in order to make the ssh server more restrictive.

Block empty passwords

PermitEmptyPasswords no

Block root user to log on to the system using ssh.

PermitRootLogin no

Limit the users allowed to access a system via SSH. In this case, only users 'laxmi' and 'vinita' are

allowed to login on the system using SSH

AllowUsers laxmi vinita

Make it more restrictive and add node address with user name. In following case only allow login through SSH users 'laxmi' and 'vinita' from 192.168.1.10 node.

AllowUsers [laxmi@192.168.1.10](mailto:laxmi@192.168.1.10) [vinita@192.168.1.10](mailto:vinita@192.168.1.10)

In addition you can restrict the access to users. In this case all users except 'user1' are allowed to

connect to the SSH server.

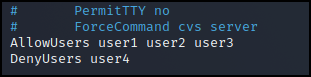
DenyUsers user1

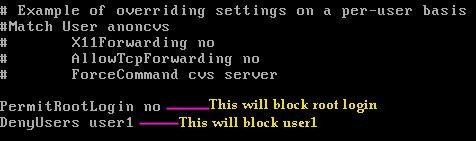
Go back on server and open main configuration file again



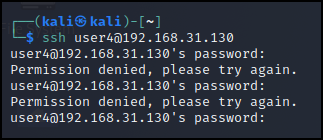
In the end of file add following directives and save the file

PermitRootLogin no DenyUsers user1





**DenyUsers user4**

****

Restart the **sshd** service



Go back to the **Linux client** system and verify that we have blocked **user1** and **root**. Also verify that **user2** able to login in SSH server.

Re-configure SSH Server to allow login only using public / private keys. Generate keys for user2 and verify that user2 can login using keys.

To make Linux servers more secure Linux administrators usually disable password authentication on the SSH server and allow only public/private keys authentication.

## **Private Keys :** Private keys are stored on the server and must be secured. Anything encrypted with the public key can only be decrypted with the paired private key. So it must be accessible only to the user owner of that key, in the **.ssh** subdirectory of that user's home directory.

## **Public Keys:** Public keys are publicly available. Public keys are required to connect with the server. The public keys for SSH servers belong to administrative workstations.

Go back on the **server** and open the main configuration file again

A screenshot of a computer

Description automatically generated with medium confidence

Uncomment following directives and save the file

RSAAuthentication yes PubkeyAuthentication yes AuthorizedKeysFile .ssh/authorized\_keys



Text

Description automatically generated

Restart the **sshd** service

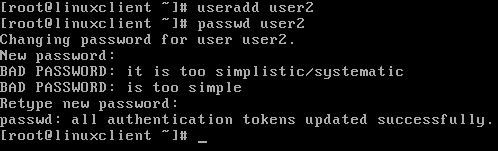


Login from **user2** and create an **ssh** directory with permission 755

A screenshot of a computer

Description automatically generated

Come back on the **linuxclient** system and create a normal user account **user2**.



A screenshot of a computer

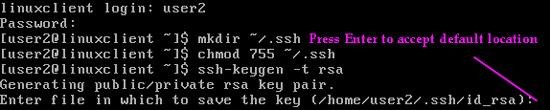
Description automatically generated with medium confidence

Login from **user2** and create an **ssh** directory with permission 755

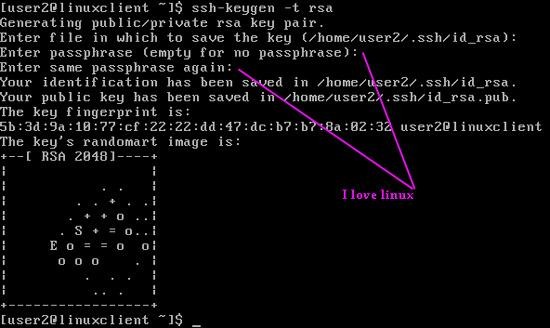
Text

Description automatically generated

Generate the public/private key pair. Accept the default location for the key file.



Enter passphrase ‘I love Linux’ and confirm



Text

Description automatically generated

The public key is stored in **/home/user2/.ssh/id\_rsa.pub**. Create a copy of the public key

Text

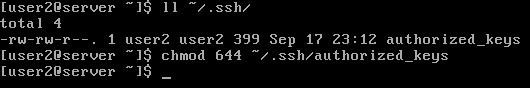
Description automatically generated

Copy the **authorized\_keys** file on server to **/home/user2/.ssh/authorized\_keys**. Enter **user2** [user account on server] password when asked

Text

Description automatically generated

On **server** verify that we have successfully copied public key on the server. Also, set permission to 644 for **authorized\_keys**



**A picture containing calendar

Description automatically generated**

Login from the **root** on server and open **sshd\_config** file

vi-sshd-config

Set **PasswordAuthentication** directive to **no** and save the file. This will block login using passwords.

A screenshot of a computer

Description automatically generated

Restart the **sshd** service

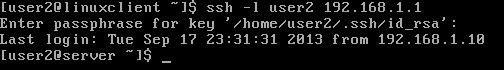
service-sshd-restart

Come back on the **Linux client** system. Logout from **user2** and login back in.

Now try to login from **user2** on **linuxclient**. Enter passphrase 'I love linux'

Text

Description automatically generated



Change default ssh port to 2223

Come on the server and open the **sshd\_config** file again

vi-sshd-config

Uncomment following directive and change value to **2223**

#port 22

Text

Description automatically generated

restart the **sshd** service

service-sshd-restart

Go back on the **Linux client** system and try to connect with the default port

Text

Description automatically generated

Now specify the new port



Text

Description automatically generated

# Post lab Questions:

1. **What is the difference between SSH & Telnet?**

**Ans:**

| **Telnet** | **SSH** |
| --- | --- |
| Telnet is the standard TCP/IP protocol for virtual terminal service. It enables you to establish a connection to a remote system in such a manner that it appears as a local system. | SSH or Secure Shell is a program to log into another computer over a network to execute commands in a remote machine. |
| Telnet uses port 23, which was designed specifically for local area networks | SSH runs on port 22 by default, which you can change it. |
| No privileges are provided for the user’s authentication. | SSH is a more secure protocol, so it uses public-key encryption for authentication. |
| Suitable for private networks | Suitable for public networks |
| Telnet transfers the data in plain text. | The encrypted format should be used to send data and also uses a secure channel. |
| Telnet is vulnerable to security attacks. | SSH helps you to overcome many security issues of Telnet. |
| Required low bandwidth usage. | Required high bandwidth usage. |
| Data sent using this protocol cannot be easily interpreted by the hackers. | Usernames and Passwords can be prone to malicious attacks. |
| Used in Linux and Windows Operating system. | All popular Operating systems. |

1. **What is SSH port forwarding?**

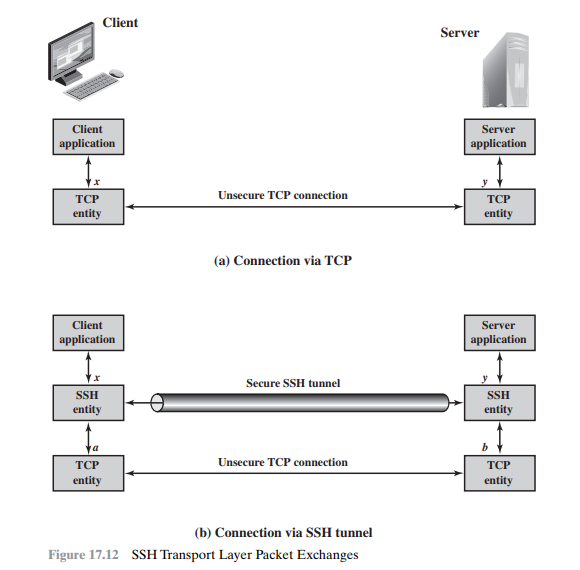
**Ans:**

One of the most useful features of SSH is port forwarding. In essence, port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. This is also referred to as SSH tunneling. We need to know what a port is in this context. A port is an identifier of a user of TCP. So, any application that runs on top of TCP has a port number. Incoming TCP traffic is delivered to the appropriate application on the basis of the port number. An application may employ multiple port numbers. For example, for the Simple Mail Transfer Protocol (smtp), the server side generally listens on port 25, so an incoming SMTP request uses TCP and addresses the data to destination port 25. TCP recognizes that this is the SMTP server address and routes the data to the SMTP server application.

Figure below illustrates the basic concept behind port forwarding. We have a client application that is identified by port number x and a server application identified by port number y. At some point, the client application invokes the local TCP entity and requests a connection to the remote server on port y. The local TCP entity negotiates a TCP connection with the remote TCP entity, such that the connection links local port x to remote port y.

To secure this connection, SSH is configured so that the SSH Transport Layer Protocol establishes a TCP connection between the SSH client and server entities, with TCP port numbers a and b, respectively. A secure SSH tunnel is established over this TCP connection. Traffic from the client at port x is redirected to the local SSH entity and travels through the tunnel where the remote SSH entity delivers the data to the server application on port y. Traffic in the other direction is similarly redirected.

SSH supports two types of port forwarding: local forwarding and remote forwarding. Local forwarding allows the client to set up a “hijacker” process. This will intercept selected application-level traffic and redirect it from an unsecured TCP connection to a secure SSH tunnel. SSH is configured to listen on selected ports. SSH grabs all traffic using a selected port and sends it through an SSH tunnel. On the other end, the SSH server sends the incoming traffic to the destination port dictated by the client application



The following example describes **local forwarding**. Suppose you have an e-mail client on your desktop and use it to get an e-mail from your mail server via the Post Office Protocol (POP). The assigned port number for POP3 is port 110. We can secure this traffic in the following way:

1. The SSH client sets up a connection to the remote server.
2. Select an unused local port number, say 9999, and configure SSH to accept traffic from this port destined for port 110 on the server.
3. The SSH client informs the SSH server to create a connection to the destination, in this case, mail server port 110.
4. The client takes any bits sent to local port 9999 and sends them to the server inside the encrypted SSH session. The SSH server decrypts the incoming bits and sends the plaintext to port 110.
5. In the other direction, the SSH server takes any bits received on port 110 and sends them inside the SSH session back to the client, who decrypts and sends them to the process connected to port 9999.

With **remote forwarding**, the user’s SSH client acts on the server’s behalf. The client receives traffic with a given destination port number, places the traffic on the correct port and sends it to the destination the user chooses. A typical example of remote forwarding is the following. You wish to access a server at work from your home computer. Because the work server is behind a firewall, it will not accept an SSH request from your home computer. However, from work, you can set up an SSH tunnel using remote forwarding. This involves the following steps.

1. From the work computer, set up an SSH connection to your home computer. The firewall will allow this because it is a protected outgoing connection.
2. Configure the SSH server to listen on a local port, say 22, and to deliver data across the SSH connection addressed to the remote port, say 2222.
3. You can now go to your home computer, and configure SSH to accept traffic on port 2222.
4. You now have an SSH tunnel that can be used for remote login to the work server.
5. **How to enable password less SSH authentication in Linux?**

**Ans:**

**SSH** (Secure Shell) allows secure remote connections between two systems. With this cryptographic protocol, you can manage machines, copy, or move files on a remote server via encrypted channels.

There are two ways to login onto a remote system over SSH – using **password authentication** or **public key authentication** (passwordless SSH login).

**In this tutorial, you will find out how to set up and enable passwordless SSH login.**

**Prerequisites**

* Access to command line/terminal window
* User with **sudo**or**root** privileges
* A local server and a remote server
* **SSH access** to a remote server via command line/terminal window

### Before You Start: Check for Existing SSH Keys

You may already have an SSH key pair generated on your machine. To see whether you have SSH keys on the system, run the command:

ls -al ~/.ssh/id\_\*.pub

If the output tells you there are no such files, move on to the next step, which shows you how to generate SSH keys.

In case you do have them, you can use the existing keys, back them up and create a new pair or overwrite it.

## **Step 1: Generate SSH Key Pair**

1. The first thing you need to do is **generate an SSH key pair** on the machine you are currently working on.

In this example, we generate a 4096-bit key pair. We also add an email address, however this is optional. The command is:

ssh-keygen -t rsa -b 4096 -C "your\_email@domain.com"

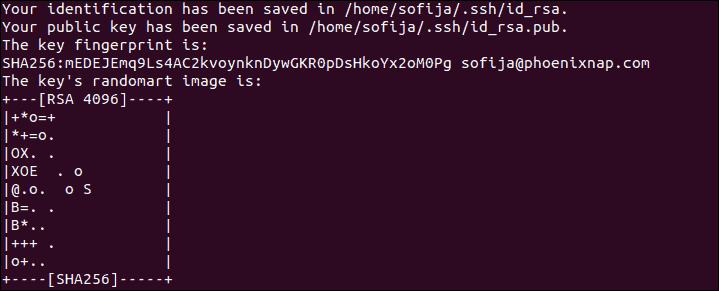
How to generate SSH key pair.

2. Next, type in the location where you want to store the keys or hit **Enter** to accept the default path.

3. It also asks you to set a passphrase. Although this makes the connection even more secure, it may interrupt when setting up automated processes. Therefore, you can type in a passphrase or just press **Enter** to skip this step.

Set a location for SSH keys and generate a password.

4. The output then tells you where it stored the identification and public key and gives you the key fingerprint.



5. Verify you have successfully created the SSH key pair by running the command:

ls -al ~/.ssh/id\_\*.pub

You should see the path of the identification key and the public key, as in the image below:

Find SSH key pair.

## **Step 2: Upload Public Key to Remote Server**

You can upload the public SSH key to a remote server with the **ssh-copy-id** command or the [**cat** command](https://phoenixnap.com/kb/linux-cat-command). Below you can find both options.

### **Option 1: Upload Public Key Using the ssh-copy-id Command**

To enable passwordless access, you need to upload a copy of the public key to the remote server.

1. Connect to the remote server and use the **ssh-copy-id** command:

ssh-copy-ide [remote\_username]@[server\_ip\_address]

2. The public key is then automatically copied into the **.ssh/authorized\_keys** file.

### **Option 2: Upload Public Key Using the cat Command**

Another way to copy the public key to the server is by using the **cat** command.

1. Start by connecting to the server and creating a **.ssh**directory on it.

ssh [remote\_username]@[server\_ip\_address] mkdir -p .ssh

2. Then, type in the password for the remote user.

3. Now you can upload the public key from the local machine to the remote server. The command also specifies that the key will be stored under the name ***authorized\_keys*** in the newly created **.ssh** directory:

cat .ssh/id\_rsa.pub | ssh [remote\_username]@[server\_ip\_address] 'cat >> .ssh/authorized\_keys'

## **Step 3: Log in to Server Without Password**

With the SSH key pair generated and the public key uploaded to the remote server, you should now be able to connect to your [dedicated server](https://phoenixnap.com/servers/dedicated) without providing a password.

Check whether the setup works by running the command:

ssh [remote\_username]@[server\_ip\_address]

The system should directly log you in to the remote server, no password required.

1. **Advantages and Disadvantages of using SSH.**

**Ans:**

## **Advantages of SSH:**

* It is available free for non-commercial use
* The open-source version has gone through improvements like bug fixes, patches, and offers many additional functionalities.
* SSH may offer multiple services using the same connection
* SSH helps you to securely tunnel insecure applications like SMTP, IMAP, POP3, and CVS.
* The tunnelling of ports works effectively for simple VPNs.
* It offers strong authentication and secure communications over insecure channels.
* SSH allows users to log into another computer over an insecure network securely.
* Provide privacy of your data via strong encryption.
* The integrity of communications performed in such a way that it cannot been altered.
* Authenticate proof of identity of senders and receivers.
* Allows you to back or forward or to encrypt other TCP/IP- based sessions.
* Allows the user to view the contents of directories, edit files, and access custom database applications remotely.

## **Disadvantage of SSH:**

* Telnet connection does not allow you to run GUI tools.
* It is not designed to transmit cursor movements or GUI movement information.
* It is not a secure protocol.
* SSH protocol not able to fix all TCP’s problems since TCP runs below SSH.
* SSH cannot protect users from attacks made through other protocols.
* This protocol does not protect Trojan horses or viruses.