CSS practical

Exp. No. 8: Configuring SSH servers on Linux Machine

What is SSH?

- SSH stand 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 internet.
- OpenSSH encrypt data before sending it over insecure network like internet.
- OpenSSH effectively eliminate eavesdropping, connection hijacking, and other attacks.
- OpenSSH provides secure tunneling and several authentication methods.
- OpenSSH replace 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

```
hardrock@hardrock-VirtualBox:~/Desktop$ sudo apt-get install openssh-server
```

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**

```
[root@server ~]# service sshd status
openssh-daemon (pid  5788) is running...
[root@server ~]# service sshd stop
Stopping sshd:                                             [  OK  ]
[root@server ~]# service sshd start
Starting sshd:                                             [  OK  ]
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# service sshd status
openssh-daemon (pid  5978) is running...
[root@server ~]# _
```

*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

```
[root@server ~]# chkconfig sshd on
[root@server ~]# _
```

IP address of OpenSSH server is required, note it down

```
[root@server ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:6F:D9:13
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe6f:d913/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1667 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1721 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:227903 (222.5 KiB)  TX bytes:254089 (248.1 KiB)

[root@server ~]# _
```

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

```
[root@linuxclient ~]# service sshd status
openssh-daemon (pid  30293) is running...
[root@linuxclient ~]# _
```

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

```
[root@linuxclient ~]# service sshd status
openssh-daemon (pid  30293) is running...
[root@linuxclient ~]# _
```

Check connectivity from SSH server

```
[root@linuxclient ~]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.974 ms
^C
--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 830ms
rtt min/avg/max/mdev = 0.974/0.974/0.974/0.000 ms
[root@linuxclient ~]# _
```

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**

```
[root@server ~]# useradd user1
[root@server ~]# useradd user2
[root@server ~]# passwd user1
Changing password for user user1.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@server ~]# passwd user2
Changing password for user user2.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@server ~]# _
```

Open main configuration file **sshd_config**

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

*NOTE: 'vi' / 'vim' is a CMD line editor.

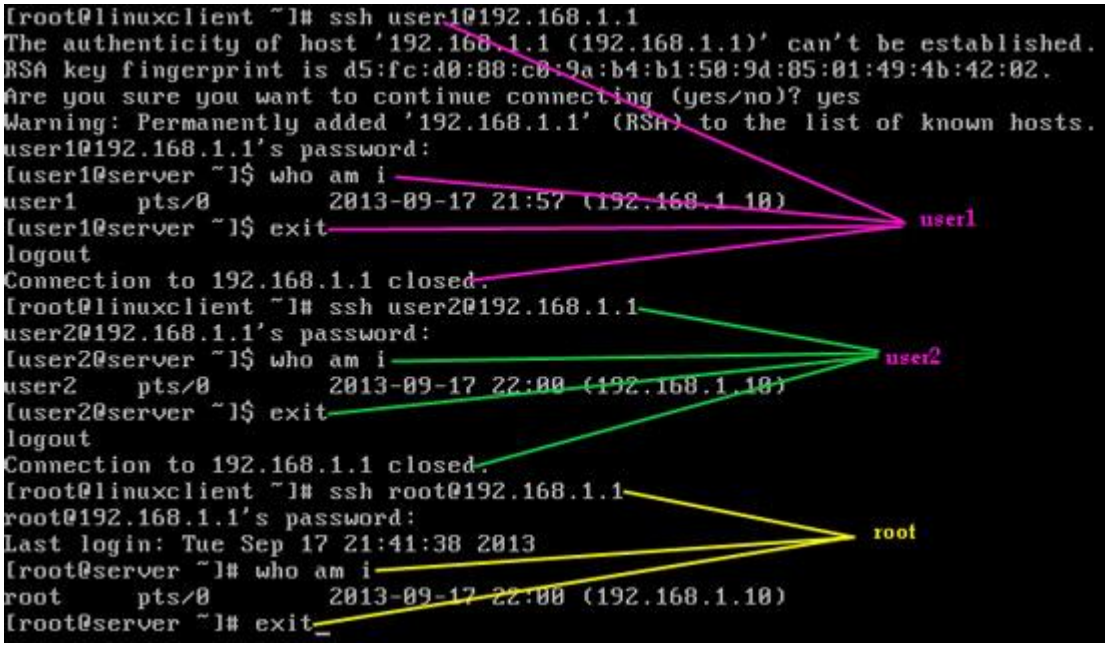
Check the value of **PasswordAuthentication** directive. In order to accept local user password base authentication it must be set to **yes**. Set it to **yes** if it is set to **no** and save the file.

```
# To disable tunneled clear text passwords,
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication yes  _
```

Restart the service if you have made any change in **sshd_config**

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

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 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 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

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

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

```
PermitRootLogin no
DenyUsers user1
```

```
# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       ForceCommand cvs server

PermitRootLogin no ——This will block root login
DenyUsers user1 ——This will block user1
```

Restart the **sshd** service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

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

```
[root@linuxclient ~]# ssh user1@192.168.1.1
user1@192.168.1.1's password:
Permission denied, please try again.          user1 and root
user1@192.168.1.1's password:                 is blocked

[root@linuxclient ~]# ssh root@192.168.1.1
root@192.168.1.1's password:
Permission denied, please try again.
root@192.168.1.1's password:

[root@linuxclient ~]# ssh user2@192.168.1.1
user2@192.168.1.1's password:
Last login: Tue Sep 17 22:00:06 2013 from 192.168.1.10
[user2@server ~]$ who am i
user2    pts/0        2013-09-17 22:35 (192.168.1.10)
[user2@server ~]$ exit
logout                                        user2 is allowed
Connection to 192.168.1.1 closed.
[root@linuxclient ~]# _
```

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 server more secure linux administrator usually disable password authentication on the SSH server and allow only public/private keys authentication.

Private Keys

Private keys are stored on server and must be secured. Anything encrypted with public key can only be decrypted with 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

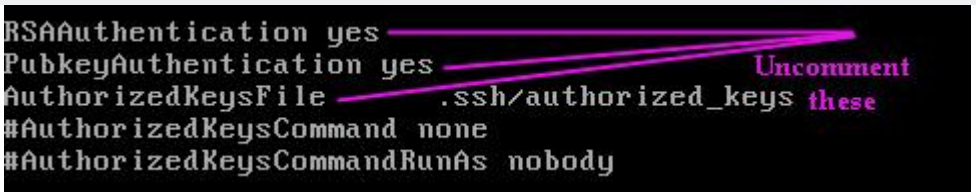
Public keys are publicly available. Public keys are required to connect with server. The public keys for SSH servers belong on administrative workstations.

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

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

Uncomment following directives and save the file

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```



Restart the **sshd** service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

Login form **user2** and create a **ssh** directory with permission 755

```
Red Hat Enterprise Linux Server release 6.1 (Santiago)
Kernel 2.6.32-131.0.15.el6.x86_64 on an x86_64

server login: user2
Password:
Last login: Tue Sep 17 22:35:07 from 192.168.1.10
[user2@server ~]$ mkdir ~/.ssh
[user2@server ~]$ chmod 755 ~/.ssh
[user2@server ~]$ _
```

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

```
[root@linuxclient ~]# useradd user2
[root@linuxclient ~]# passwd user2
Changing password for user user2.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@linuxclient ~]# _
```

Login form **user2** and create a **ssh** directory with permission 755

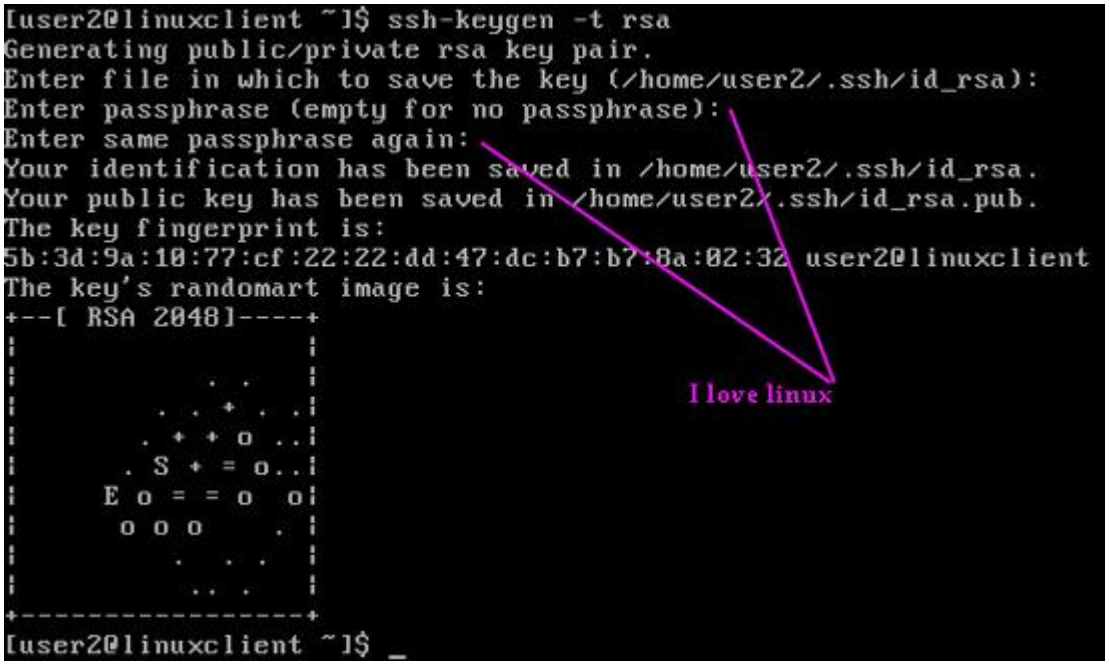
```
Red Hat Enterprise Linux Server release 6.1 (Santiago)
Kernel 2.6.32-131.0.15.el6.x86_64 on an x86_64

linuxclient login: user2
Password:
[user2@linuxclient ~]$ mkdir ~/.ssh
[user2@linuxclient ~]$ chmod 755 ~/.ssh
[user2@linuxclient ~]$ _
```

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

```
linuxclient login: user2
Password:
[user2@linuxclient ~]$ mkdir ~/.ssh Press Enter to accept default location
[user2@linuxclient ~]$ chmod 755 ~/.ssh
[user2@linuxclient ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user2/.ssh/id_rsa):
```

Enter passphrase 'I love linux' and confirm



Public key is stored in **/home/user2/.ssh/id_rsa.pub**. Create a copy of public key

```
[user2@linuxclient ~]$ cat ~/.ssh/id_rsa.pub >> authorized_keys
[user2@linuxclient ~]$ _
```

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

```
[user2@linuxclient ~]$ scp authorized_keys user2@192.168.1.1:/home/user2/.ssh/
```

On **server** verify that we have successfully copied public key on server. Also set permission to 644 for **authorized_keys**

```
[user2@server ~]$ ll ~/.ssh/
total 4
-rw-rw-r--. 1 user2 user2 399 Sep 17 23:12 authorized_keys
[user2@server ~]$ chmod 644 ~/.ssh/authorized_keys
[user2@server ~]$ _
```

Login from **root** on server and open **sshd_config** file

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

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

```
# To disable tunneled clear text passwords,
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no
```

Restart the **sshd** service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]#
```

Come back on **linuxclient** system.

Logout from **user2** and login back.

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

```
[user2@linuxclient ~]$ ssh -l user2 192.168.1.1
Enter passphrase for key '/home/user2/.ssh/id_rsa':
Last login: Tue Sep 17 23:31:31 2013 from 192.168.1.10
[user2@server ~]$
```

Change default ssh port to 2223
Come on server and open **sshd_config** file again

```
[root@server ~]# vi /etc/ssh/sshd_config
```

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

```
#port 22
# The strategy used for options in the default sshd_config
# OpenSSH is to specify options with their default value wh
# possible, but leave them commented.  Uncommented options
# default value.

Port 2223 ———— Uncomment and change it to 2223
```

restart the **sshd** service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]#
```

Go back on **linuxclient** system and try to connect with default port

```
[user2@linuxclient ~]$ ssh -l user2 192.168.1.1
ssh: connect to host 192.168.1.1 port 22: Connection refused
[user2@linuxclient ~]$ _
```

Now specify the new port

Postlab Questions:

1. What is the difference between ssh & Telnet ?
2. What is SSH port forwarding?
3. How to enable passwordless ssh authentication in Linux ?
4. Advantages and Disadvantages of using SSH.