SSH is a software package that enables system administrators to securely administer their systems from a remote network with user authentication, commands, output, and file transfers being encrypted to protect against attacks in the network. While it is much more secure than common login services such as telnet, ftp, rlogin, rsh, and rcp, its default installation isn't hardened, following simple steps can dramatically harden an installation.

- Firstly make sure to keep ssh software package up to date.

```
# yum update openssh
```
(or any other ssh package)

Note: SSH configuration files are stored in the **/etc/ssh/sshd_config** file. Once all configuration changes have been made , make sure to restart sshd

```
# service sshd restart
```

1. Set proper permission for sshd configuration file, to restrict non-privileged users from making unauthorized changes to the configuration file since it controls , who, how and from where one can remotely access the server.

```
# chown root:root /etc/ssh/sshd_config

# chmod og-rwx /etc/ssh/sshd_config
```

2. **Initial setup**

Use Strong Passwords

If you monitor your ssh port 22 (the default port on which ssh listens) that is exposed to outside world you will notice that there are thousands of bruteforce attempts per day. With strong passwords in place, you would will have enough time to notice bruteforce attempts through logs before they actually manage to guess them and get into your system.

If you are already not using a strong password, try to choose passwords that contains:

- Minimum of 8 characters
- Mix of upper and lower case letters
- Mix of letters and numbers
- Non alphanumeric characters (e.g. special characters such as ! " £ $ % ^ etc)
- Best is to choose a phrase and mix above characters
- Difficult to remember too many complex passwords? Use password manager.

Strong password isn't specific to ssh, for more details please visit :www.btcirt.bt/advisory

a. Add new user and set strong password :

```
# adduser user1
# passwd  user1
```

b. Provide root privilege:

```
# gpasswd -a user1 wheel
```

c. Create Public/Private Key Pair in local machine (run following commands in mac/Linux or use PuttyGen for windows machine). Make sure to set passphrase.

```
# ssh-keygen
# cat ~/.ssh/id_rsa.pub
```

Copy and paste the public key to some notepads.

d. Login back to the server via ssh as root user and do the following:

```
#  su - user1
#  mkdir .ssh
#  chmod 700 .ssh
#  vi .ssh/authorized_keys
#  chmod 600 .ssh/authorized_keys
#  exit    (return to root)
```

(paste the public key here)

3. **Disable remote access to the root account:**
Restricting ssh root login, forces system admins to authenticate using their own individual account, then escalating to root via sudo or su . This in turn limits opportunity for non-repudiation and provides a clear audit trail in the event of a security incident.

```
#  vi /etc/ssh/sshd_config
```

PermitRootLogin no

4. **Enable Public key authentication:**
Since you had already generated and copied public key to server during initial setup now enable public/private key authenticate method.

```
#  vi /etc/ssh/sshd_config
```

RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys

5. **Disable Password login:** Now that we are using pub keys to authenticate and connect to server make sure to restrict users from login to the server using passwords, else you would still be vulnerable to brute force attacks.

```
#  vi /etc/ssh/sshd_config
```

PasswordAuthentication no

6. **Set SSH Protocol to 2:**

SSH supports two different and incompatible protocols: SSH1 and SSH2. The older protocol 1 is less secure and should be disabled unless there is specific requirement. SSH2 is more advanced and secure. Unless you explicitly specify ssh protocol , protocol 1 is enabled by default.

```
#  vi /etc/ssh/sshd_config
```

Protocol 2

7. **Set SSH LogLevel to INFO:**

When an incident occurs, it is important to determine when a particular user, logged in and out of the system. All login and logout activities will be logged when LogLevel is set to Info .

```
# vi /etc/ssh/sshd_config
```

LogLevel INFO

8. **Ensure SSH X11 forwarding is disabled**

X11 forwarding is enabled when you need to use some tool that has a GUI remotely. It allows a user to start up remote applications but forward the application display to your local machine. Thus, it can permit a malicious user to secretly open another X11 connection to another remote client during the session compromised it.

```
# vi /etc/ssh/sshd_config
    X11Forwarding no
```

9. **Ensure SSH MaxAuthTries is set to 4 or less**

Value of MaxAuthTries will limit the number of unsuccessful attempts a user can make thus minimizing the risk of bruteforce attack. Set this value to as minimum as possible, recommended is 4, but it can be set to lower value too.

```
# vi /etc/ssh/sshd_config
    MaxAuthTries 4
```

10. **Ensure SSH IgnoreRhosts is enabled**

Setting this parameter forces users to enter a password when authenticating with ssh.

```
# vi /etc/ssh/sshd_config
    IgnoreRhosts yes
```

11. **Ensure SSH HostbasedAuthentication is disabled**

Host-based authentication uses the public host key of the client machine to authenticate a user to the remote server, server relies on the integrity of the user account management on the client host. This provides a non-interactive form of authentication, and is best used in scripts and automated processes, such as cron jobs. It is not recommended that hosts unilaterally trust one another , even within an organization.

```
# vi /etc/ssh/sshd_config
    HostbasedAuthentication no
```

12. **Ensure SSH PermitEmptyPasswords is disabled**

Restrict access to server via ssh for accounts with no password, thus minimizing the probability of unauthorized access .

```
# vi /etc/ssh/sshd_config

    PermitEmptyPasswords no
```

13. **Ensure SSH PermitUserEnvironment is disabled**

By restricting users from setting environment variables through ssh, you can reduce the risk of user bypassing security controls and setting harmful execution path for instance.

```
# vi /etc/ssh/sshd_config
        PermitUserEnvironment no
```

14. Use only approved MAC algorithms and Ciphers

Weaks MACs(Message Authentication Code) and Ciphers can be easily broken or exploited. Therefore, It is recommended to enable support for only strong MACs and ciphers.

```
# vi /etc/ssh/sshd_config
```

a. Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr

b. MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com

15. Ensure SSH Idle Timeout Interval is configured

Most admins accidentally leave their remote sessions logged in, with timeout value set, remote session will automatically close if the user isn't active for the set interval, thus reducing the risk of unauthorized user access to your ssh session.

```
# vi /etc/ssh/sshd_config
        ClientAliveInterval 300
        ClientAliveCountMax 0
```

16. Ensure SSH LoginGraceTime is set to one minute or less

The LoginGraceTime parameter specifies the time allowed for successful authentication to the SSH server. The longer the Grace period is the more open unauthenticated connections can exist. Set it to as short as possible.

```
# vi /etc/ssh/sshd_config
        LoginGraceTime 60
```

17. Ensure SSH access is limited

Limit who can ssh to your server with AllowUsers or AllowGroups variable followed by a space separated list of authorized usernames or groups respectively, thus denying any unauthorized user or groups from accessing the server.

```
# vi /etc/ssh/sshd_config
        AllowUsers user1  user2
              Or

        AllowGroups sshusers
```

18. Ensure SSH warning banner is configured

While setting banner message isn't actually related to securing the remote server but will come in handy during the time of procession, if your system is compromised.

```
# vi /etc/ssh/sshd_config
```

Banner /etc/issue.net

Example banner message :

```
# vi /etc/issue.net
```

This system is owned by ABC company and is for authorized use only. All activities are recorded and monitored

19. Use a Non-Standard Port

By default, ssh listens for incoming connections on port 22, which is being scanned for every seconds by attackers. Thus, it is recommended to choose any random port above 1024, that's not used for any known services and ofcourse not 2222, which is already known to be alternate for 22. For example :

```
# vi /etc/ssh/sshd_config
```

Port 2431

According make changes to your iptables or firewalld selinux, port forwarding in your router and any other applicable firewall rules. For example

If firewalld is enabled :

```
# firewall-cmd --add-port 2431/tcp
# firewall-cmd --add-port 2431/tcp --permanent
```

If iptables is used :

```
# iptables -I INPUT -p tcp --dport 2431 -j ACCEPT
```

If selinux is enabled

```
# semanage port -a -t ssh_port_t -p tcp 2431
```

Now, while connecting to server using ssh from client you need to specify new port number, since the standard port is no more used. For example:

```
# ssh user@servername  -p 2431
```

20. Filter SSH at the Firewall or with TCP Wrappers:

Restrict ssh access to only authorized ip addresses by restricting it with firewall rules and by appropriately configuring hosts.allow and hosts.deny

iptables:

```
# iptables -A INPUT -p tcp -s 172.20.3.15 --dport 22 -j ACCEPT
```

Or with TCP Wrappers

```
# vi  hosts.deny
```

ALL : ALL

```
# vi hosts.allow
```

ALL : 172.20.3.15

Reference:

1. https://www.cisecurity.org/benchmark/centos_linux/
2. https://linux-audit.com/audit-and-harden-your-ssh-configuration/