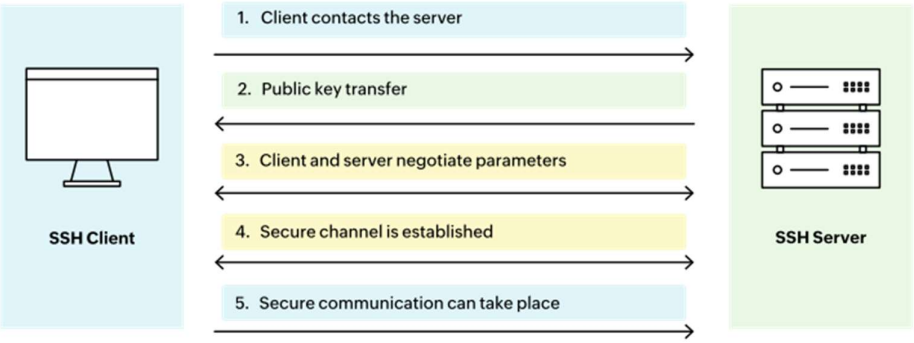
CHAPTER-10 CONFIGURE AND SECURE SSH

Access the Remote Command Line with SSH

Secure Shell

The **OpenSSH package** provides **the Secure Shell or SSH protocol** in Red Hat Enterprise Linux. With SSH protocol, systems can communicate in an encrypted and secure channel over an insecure network. Use the **ssh command to create a secure connection** to **a remote system**, authenticate as a specific user, and obtain an interactive shell session on the remote system. The ssh command can run a session on a remote system without running an interactive shell.



Connect with Secure Shell

The following ssh command logs you in on the remote server using the same username as the current local user.

DESCRIPTION	COMMANDS / OPTIONS
Login to remote host using ssh	**Syntax:** ssh [options][username]@[hostname/IP address] -1 Forces ssh to use protocol SSH-1 only. -2 Forces ssh to use protocol SSH-2 only. -4 Allows IPv4 addresses only. -6 Allows IPv6 addresses only. -A Authentication agent connection forwarding is enabled. -a Authentication agent connection forwarding is disabled. -C Compresses all data -c Selects the cipher specification for encrypting the session -g Allows remote hosts to connect to local forwarded ports. -p Port to connect to on the remote host. **Example:** [root@host ~]# ssh hosta@remote_host↵
Identifying Remote Users	**Syntax:** w [options] user **-h** Suppresses the header row from being displayed in the output.

	-u Ignores the username when calculating the current process and CPU times. **-s** Uses the short format **-f** Toggles the printing of the 'from' field (remote hostname). By default, it is not printed, but this option can change that. **-I** Displays the IP address instead of the hostname **-V** Displays version information about the 'w' command. **-o** Prints a blank space for idle times that are less than one minute.

SSH Host Keys

SSH **secures communication through public-key encryption**. When an **SSH client connects to an SSH server**, the **server sends a copy** of its **public key** to the **client before logging in**. This key helps **to set up secure encryption** for the communication channel and to **authenticate the client'ssystem**.
When a **user** runs the ssh command for connecting **to an SSH server**, the command **checks for a copy of the server's public key** in its local known **hosts file**. The key might be **preconfigured in the /etc/ssh/ssh_known_hosts** file, or the user might have the **~/.ssh/known_hosts file** that contains the key **in their home directory**.

If the **client has a copy of the key**, then the **ssh command compares the key from the known hosts server files** to **the key that it received**. If the keys do not match, then the client assumes that the network traffic to the server is compromised and prompts the user to confirm whether to continue with the connection.

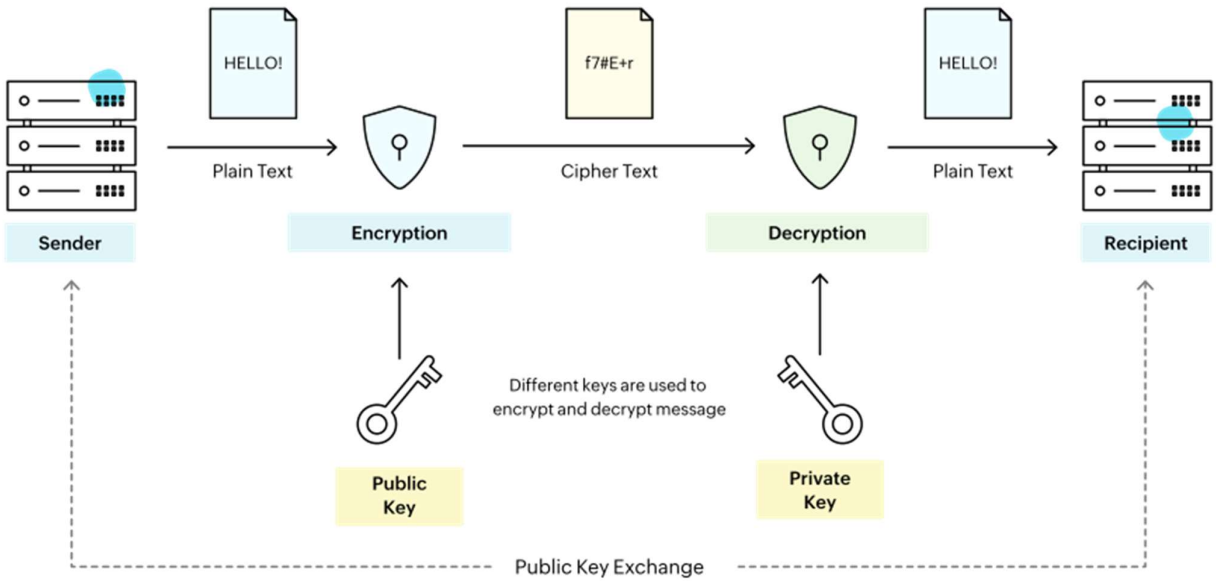
Strict Host Key Checking

The **StrictHostKeyChecking** parameter is set in the user-specific **~/.ssh/config file**, or in the system-wide **/etc/ssh/ssh_config file**, or by specifying **the –o StrictHostKeyChecking= option** of the ssh command.

- If the **StrictHostKeyChecking** parameter is set to yes, then the ssh command always aborts the SSH connection if the public keys do not match.
- If the **StrictHostKeyChecking** parameter is set to no, then the ssh command enables the connection and adds the key of the target host to the **~/.ssh/known_hosts** file.
- If the target host SSH key changed since the last time that you connected to it, then the ssh command asks for confirmation to log in and accept the new key

- If you accept the new key, then a copy of the public key is saved in the **~/.ssh/known_hosts** file to automatically confirm the server's identity on subsequent connections.

DESCRIPTIN	COMMANDS / OPTIONS
To Verify the fingerprint of the target server's SSH host key	**Syntax:** ssh-keygen -lf [key Location in /etc] **Example:** [user@host ~]$ ssh-keygen –lf /etc/ssh/ssh _host _ecdsa _key.pub ↵



SSH Known Hosts Key Management

Information about known remote systems and their keys are stored in either:

- The system-wide /etc/ssh/ssh_known_hosts file.
- The ~/.ssh/known_hosts file in each user's home directory.

The **/etc/ssh/ssh_known_hosts** file is a system-wide file that **stores the public keys** for *hosts known* by the system. You must create and manage this file, either manually or through some automated method such as using Ansible or a script that **uses the ssh-keyscan** utility.

If a server's **public key is changed** because the key was lost due to hard drive failure or it was replaced for some legitimate reason, then for successful log in to that system, the **/etc/ssh/ssh_known_hosts** file must be modified **to replace the previous public key entry** with the **new public key**.

If you connect to a remote system and the public key of that system is not in **the /etc/ssh/ ssh_known_hosts** file, then the SSH client searches for the key in **the ~/.ssh/known_hosts** file.

Each known host key entry consists of one line containing three fields:

- The first field is the list of hostnames and IP addresses that share the public key.
- The second field is the encryption algorithm that is used for the key.
- The last field is the key itself.

Configure SSH Key-based Authentication

Configure your account for passwordless access to SSH servers that have enabled keybased authentication, which is based on public key encryption (PKI).

DESCRIPTION	COMMANDS / OPTIONS

authentication key generation, management and conversion	**Syntax:** ssh-keygen [options]
	To create a key pair
	Example: [user@host ~]$ ssh-keygen ↵
	`[user@host ~]$ ssh-keygen`
	`Generating public/private rsa key pair.`
	`Enter file in which to save the key (/home/user/.ssh/id_rsa): Enter`
	`Created directory '/home/user/.ssh'.`
	`Enter passphrase (empty for no passphrase): Enter`
	`Enter same passphrase again: Enter`
	
	A passphrase-protected private key is created with the public key
	Example: [user@host ~]$ ssh-keygen -f .ssh/key-with-pass ↵
	`[user@host ~]$ ssh-keygen -f .ssh/key-with-pass`
	`Generating public/private rsa key pair.`
	`Enter passphrase (empty for no passphrase): your_passphrase`
	`Enter same passphrase again: your_passphrase`
	
	Share the Public Key
	To configure your remote account for access, copy your public key to the remote system. The ssh-copy-id command copies the public key of the SSH key pair to the remote system. You can specify a specific public key with the ssh-copy-id command, or use the default ~/.ssh/ id_rsa.pub file.
	Example: [user@host] ssh-copy-id -i .ssh/key-with-pass.pub user@remotehost ↵

SSH Client Configuration

You can **create the ~/.ssh/config file** to **preconfigure SSH connections**. Within the configuration file, you can specify connection parameters such as users, keys, and ports for specific hosts. This file eliminates the need to manually specify command parameters each time that you connect to a host. Consider the following ~/.ssh/config file, which preconfigures two host connections with different users and keys:

```
[user@host ~]$ cat ~/.ssh/config
host servera
     HostName                    servera.example.com
     User                        usera
     IdentityFile                ~/.ssh/id_rsa_servera

host serverb
     HostName                    serverb.example.com
     User                        userb
     IdentityFile                ~/.ssh/id_rsa_serverb
```

The **~/.ssh/config** file is also useful for configuring SSH jump hosts. An **SSH jump host** is a **server that acts as a proxy for SSH connections to other**, usually **internal, hosts**. Consider a scenario where a **host called external** is accessible via **the internet**, but a host called **internal is only internally accessible**. Use the ProxyHost parameter within the ~/.ssh/config file to connect to the internal host via the external host:

```
[user@host ~]$ cat ~/.ssh/config
host internal
     HostName                    internal.example.com
     ProxyHost                   external

host external
     HostName                    external.example.com
```

Customize OpenSSH Service Configuration

Configure the OpenSSH Server

The sshd daemon provides the OpenSSH service. You can configure the service by editing the /etc/ssh/sshd_config file.

Prohibit the Superuser from Logging In

It is a good practice to prohibit direct login to the root user account from remote systems. Some of the risks of allowing direct login as the root user include the following cases:

- The root user name exists on every Linux system by default, so a potential attacker needs only to guess the password, instead of a valid user name and password combination. This scenario reduces complexity for an attacker.
- The root user has unrestricted privileges, so its compromise can lead to maximum damage to the system.
- From an auditing perspective, it can be hard to track which authorized user logged in as the root user and made changes. If users have to log in as a regular user and switch to the root account, then you can view a log event to help to provide accountability.

The OpenSSH server uses the **PermitRootLogin** configuration setting in **the /etc/ssh/ sshd_config file** to **allow or prohibit users** to log in to the system as root. With the **PermitRootLogin parameter set to yes**, as it is by default, people are permitted to log in as the root user. To **prevent this situation, set the value to no**.

```
PermitRootLogin yes
```

The SSH server (sshd) must be reloaded for any changes to take effect.

```
[root@host ~]# systemctl reload sshd
```

Prohibit Password-based Authentication for SSH

The OpenSSH server uses the **PasswordAuthentication** parameter in the **/etc/ssh/ sshd_config file** to control whether users can use password-based authentication to log in to the system.

```
PasswordAuthentication yes
```

DESCRIPTION	COMMANDS / OPTIONS
Secure copy	**Syntax:** scp [option] <source ><RemoteHost>: <RmoteLocation> **-P** port: Specifies the port to connect on the remote host **-p** Preserves modification times, access times, and modes from the original file **-q** Disables the progress meter **-r** Recursively copy entire directories **-s** Name of program to use for the encrypted connection