Oracle® Linux

Connecting to Remote Systems With OpenSSH
Abstract

*Oracle® Linux: Connecting to Remote Systems With OpenSSH* describes how to configure and use OpenSSH to securely connect to systems on the network.

Document generated on: 2020-06-08 (revision: 10129)
Table of Contents

Preface ................................................................................................................................................................. v
1 About OpenSSH .................................................................................................................................................. 1
2 About OpenSSH Configuration .......................................................................................................................... 3
   2.1 OpenSSH User Configuration Files ........................................................................................................... 3
   2.1.1 User Configuration Files in ~/.ssh on the Client ................................................................................. 3
   2.1.2 User Configuration Files in ~/.ssh on the Server ............................................................................... 4
3 Installing and Configuring an OpenSSH Server and Client .......................................................................... 5
   3.1 Installing and Configuring an OpenSSH Server ....................................................................................... 5
   3.2 Installing OpenSSH Client Packages ..................................................................................................... 5
4 Administering OpenSSH .................................................................................................................................... 7
   4.1 Using OpenSSH Utilities .......................................................................................................................... 7
   4.1.1 Using the ssh Command to Connect to Another System ................................................................. 8
   4.1.2 Using the scp and sftp Commands to Copy Files Between Systems ............................................. 8
   4.1.3 Generating Pairs of Authentication Keys by Using the ssh-keygen Command ........................... 9
   4.1.4 Enabling Remote System Access Without Requiring a Password .................................................. 10
   4.2 Restricting Access to SSH Connections ................................................................................................. 12
Preface

Oracle® Linux: Connecting to Remote Systems With OpenSSH describes how to configure the OpenSSH feature and use it to connect to remote systems.

Note

This guide contains content that was tested against Oracle Linux 8, but generally applies to most Oracle Linux releases and may also apply to other distributions.

Audience

This document is intended for administrators who need to configure and administer Oracle Linux features. It is assumed that readers are familiar with web technologies and have a general understanding of using the Linux operating system, including knowledge of how to use a text editor such as emacs or vim, essential commands such as cd, chmod, chown, ls, mkdir, mv, ps, pwd, and rm, and using the man command to view manual pages.

Document Organization

The document is organized into the following chapters:

• Chapter 1, About OpenSSH describes the Oracle Linux describes the OpenSSH technology.

• Chapter 2, About OpenSSH Configuration describes the configuration files that are used by OpenSSH.

• Chapter 3, Installing and Configuring an OpenSSH Server and Client describes how to install and configure an OpenSSH server and includes instructions for installing the OpenSSH client packages.

• Chapter 4, Administering OpenSSH describes how to use OpenSSH utilities and includes other administrative tasks.

Related Documents

The documentation for this product is available at:


Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.
Chapter 1 About OpenSSH

This chapter provides an overview of the OpenSSH technology, which is used to support secure communications between networked systems.

Note

This document includes content that was tested against Oracle Linux 8, but generally applies to most Oracle Linux releases, and may also apply to other distributions.

OpenSSH is suite of network connectivity tools that provides secure communications between systems. The tools that are part of the OpenSSH suite include the following:

- **scp**: Secure file copying.
- **sftp**: Secure File Transfer Protocol (FTP).
- **ssh**: Secure shell to log on to or run a command on a remote system.
- **sshd**: Daemon that supports the OpenSSH services.
- **ssh-keygen**: Creates RSA authentication keys.

Note

The Digital Signature Algorithm (DSA) is considered deprecated in this release. As such, authentication mechanisms that depend on DSA keys do not work in the default configuration. Note also that OpenSSH clients do not accept DSA host keys, even at the LEGACY system-wide cryptographic policy level.

Unlike utilities such as `rcp`, `ftp`, `telnet`, `rsh`, and `rlogin`, OpenSSH tools encrypt all of the network packets between the client and server, including password authentication.

OpenSSH supports the SSH version 2 (SSH2) protocol. In addition, OpenSSH provides a secure way of using graphical applications over a network by using X11 forwarding. OpenSSH also use port forwarding as another way to secure otherwise insecure TCP/IP protocols.
Chapter 2 About OpenSSH Configuration

Table of Contents

2.1 OpenSSH User Configuration Files ................................................................. 3
  2.1.1 User Configuration Files in ~/.ssh on the Client ......................................... 3
  2.1.2 User Configuration Files in ~/.ssh on the Server ......................................... 4

This chapter describes OpenSSH configuration files, including detailed information about OpenSSH user configuration files.

The following OpenSSH global configuration files are located in the /etc/ssh directory:

moduli                  Contains key-exchange information that is used to set up a secure connection.

ssh_config              Contains default client configuration settings that can be overridden by the settings in a user's ~/.ssh/config file.

ssh_host_rsa_key         Contains the RSA private key for SSH2.

ssh_host_rsa_key.pub     Contains the RSA public key for SSH2.

sshd_config             Contains configuration settings for sshd.

Other files can be configured in this directory. For details, see the sshd(8) manual page.

For more information, see the ssh_config(5), sshd(8), and sshd_config(5) manual pages.

2.1 OpenSSH User Configuration Files

To use the OpenSSH tools, a user must have an account on both the client and server systems. Note that these accounts do not need to be configured identically on each system.

User configuration files are located in the .ssh directory in a user's home directory (~/.ssh) on both the client and server. OpenSSH creates this directory and the known_hosts file when the user first uses an OpenSSH utility to connect to a remote system.

2.1.1 User Configuration Files in ~/.ssh on the Client

On the client side, the ~/.ssh/known_hosts file contains the public host keys that OpenSSH has obtained from SSH servers. OpenSSH adds an entry for each new server to which a user connects.

In addition, the ~/.ssh directory usually contains one of the following pairs of key files:

id_rsa and id_rsa.pub    Contains a user's SSH2 RSA private and public keys. SSH2 RSA is most commonly used key-pair type.

Caution

The private key file can be readable and writable by the user but must not be accessible to other users.

The optional config file contains client configuration settings.
User Configuration Files in ~/.ssh on the Server

2.1.2 User Configuration Files in ~/.ssh on the Server

On the server side, the ~/.ssh directory usually contains the following files:

- **authorized_keys**: Contains your authorized public keys. The server uses the signed public key in this file to authenticate a client.

- **config**: Contains client configuration settings. This file is optional.

  **Caution**
  
  A config file can be readable and writable by the user but must not be accessible to other users.

- **environment**: Contains definitions of environment variables. This file is optional.

- **rc**: Contains commands that ssh executes when a user logs in, before the user’s shell or command runs. This file is optional.

  **Caution**
  
  A config file can be readable and writable by the user but must not be accessible to other users.

For more information, see the ssh(1) and ssh-keygen(1) manual pages.
Chapter 3 Installing and Configuring an OpenSSH Server and Client

Table of Contents

3.1 Installing and Configuring an OpenSSH Server ................................................................. 5
3.2 Installing OpenSSH Client Packages .................................................................................. 5

This chapter describes how to install and configure an OpenSSH server. Information about installing OpenSSH client packages is also provided.

3.1 Installing and Configuring an OpenSSH Server

A default Oracle Linux installation includes the openssh and openssh-server packages, but the sshd service is not enabled by default.

To configure an OpenSSH server:

1. If necessary, install or update the openssh and openssh-server packages:

   # dnf install openssh openssh-server

2. Start the sshd service and configure it to start following a system reboot:

   # systemctl start sshd
   # systemctl enable sshd

   You can set sshd configuration options for features such as Kerberos authentication, X11 forwarding, and port forwarding in the /etc/ssh/sshd_config file.

   For more information, see the sshd(8) and sshd_config(5) manual pages.

3.2 Installing OpenSSH Client Packages

To configure OpenSSH, you need the openssh and openssh-clients packages. A default Oracle Linux installation includes both of these packages. However, if necessary, install or update the openssh and openssh-clients packages on your system as follows:

   # dnf install openssh openssh-clients
Chapter 4 Administering OpenSSH

Table of Contents

4.1 Using OpenSSH Utilities ................................................................. 7
  4.1.1 Using the ssh Command to Connect to Another System ................... 8
  4.1.2 Using the scp and sftp Commands to Copy Files Between Systems .......... 8
  4.1.3 Generating Pairs of Authentication Keys by Using the ssh-keygen Command 9
  4.1.4 Enabling Remote System Access Without Requiring a Password .......... 10
4.2 Restricting Access to SSH Connections ........................................ 12

This chapter provides information about administering OpenSSH by using OpenSSH utilities. The chapter also explains how to restrict access to SSH connections.

4.1 Using OpenSSH Utilities

By default, each time you use the OpenSSH utilities to connect to a remote system, you must provide your user name and password. When you connect to an OpenSSH server for the first time, the OpenSSH client prompts you to confirm that you are connected to the correct system.

The following examples show how you would connect to a remote host, host04:

```
$ ssh host04
The authenticity of host 'host04 (192.0.2.104)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'host04,192.0.2.104' (RSA) to the list of known hosts.
```

When you type yes to accept the connection to the server, the client adds the server’s public host key to the your ~/.ssh/known_hosts file. When you next connect to the remote server, the client compares the key in this file to the one that the server supplies. If the keys do not match, you see a warning such as the following:

```
@ WARNING: POSSIBLE DNS SPOOFING DETECTED!  @
The RSA host key for host has changed, and the key for the according IP address IP_address is unchanged. This could either mean that DNS SPOOFING is happening or the IP address for the host and its host key have changed at the same time.
Offending key for IP in /home/user/.ssh/known_hosts:10
WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!  @
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)! It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is fingerprint
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of this message.
Offending key in /home/user/.ssh/known_hosts:53
RSA host key for host has changed and you have requested strict checking.
Host key verification failed.
```
Unless there is a reason for the remote server’s host key to have changed, such as an upgrade of either the SSH software or the server, you should not try to connect to that machine until you have contacted its administrator about the situation.

### 4.1.1 Using the `ssh` Command to Connect to Another System

Use the `ssh` command to log into a remote system or to execute a command on a remote system:

```bash
$ ssh [options] [user@]host [command]
```

where `host` is the name of the remote OpenSSH server to which you want to connect.

For example, you would log in to `host04` by using the same user name as is on the local system:

```bash
$ ssh host04
```

The remote system prompts you for your password on that system.

To connect as a different user, specify the user name and `@` symbol before the remote host name, for example:

```bash
$ ssh joe@host04
```

To execute a command on the remote system, specify the command as an argument:

```bash
$ ssh joe@host04 ls ~/.ssh
```

The `ssh` command logs you in, executes the command, and then closes the connection.

For more information, see the `ssh(1)` manual page.

### 4.1.2 Using the `scp` and `sftp` Commands to Copy Files Between Systems

The `scp` command enables you to copy files or directories between systems. `scp` establishes a connection, copies the files, and then closes the connection.

To upload a local file to a remote system:

```bash
$ scp [options] local_file [user@]host[:remote_file]
```

For example, copy `testfile` to your home directory on `host04`:

```bash
$ scp testfile host04
```

Copy `testfile` to the same directory but change its name to `new_testfile`:

```bash
$ scp testfile host04:new_testfile
```

To download a file from a remote system to the local system:

```bash
$ scp [options] [user@]host[:remote_file] local_file
```

The `-r` option enables you to recursively copy the contents of directories. For example, copy the directory `remdir` and its contents from your home directory on remote `host04` to your local home directory:

```bash
$ scp -r host04:~/remdir ~
```
The `sftp` command is a secure alternative to the `ftp` command that is used to transfer files between systems. Unlike the `scp` command, the `sftp` command enables you to browse the file system on the remote server before copying any files.

To open an FTP connection to a remote system over SSH, use the following command:

```
$ sftp [options] [user@]host
```

For example, you would open an FTP connect to the system, `host04`, as follows:

```
$ sftp host04
Connecting to host04...
guest@host04's password: password
sftp>
```

You type `sftp` commands at the `sftp>` prompt.

In the following example, the `put` command is used to upload the file `newfile` from the local system to the the remote system, then the `ls` command is used to list it:

```
sftp> put newfile
Uploading newfile to /home/guest/newfile
foo                                           100%   1198  1.2KB/s   00:01
sftp>
ls foo
foo
```

Type `help` or `?` to display a list of available commands. Type `bye`, `exit`, or `quit` to close the connection and exit the `sftp` interactive session.

For more information, see the `ssh(1)` and `sftp(1)` manual pages.

### 4.1.3 Generating Pairs of Authentication Keys by Using the `ssh-keygen` Command

The `ssh-keygen` command generate a public and private authentication key pair. Such authentication keys allow you to connect to a remote system without needing to supply a password each time that you connect. Each user must generate their own pair of keys. If `root` generates key pairs, only `root` can use those keys.

To create a public and private SSH2 RSA key pair:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/guest/.ssh/id_rsa): <Enter>
Created directory '/home/guest/.ssh'.
Enter passphrase (empty for no passphrase): password
Enter same passphrase again: password
Your identification has been saved in /home/guest/.ssh/id_rsa.
Your public key has been saved in /home/guest/.ssh/id_rsa.pub.
The key fingerprint is: 5e:d2:66:f4:2c:c5:cc:07:92:97:c9:30:0b:11:90:59 guest@host01
The key's randomart image is:
```
```
```
Enabling Remote System Access Without Requiring a Password

To create an SSH key pair by using an algorithm other than the default RSA algorithm, use the `-t` option. Possible values that you can specify include the following: `dsa`, `ecdsa`, `ed25519`, and `rsa`.

For security, in case an attacker gains access to your private key, you can specify an passphrase to encrypt your private key. If you encrypt your private key, you must enter this passphrase each time that you use the key. If you do not specify a passphrase, you are not prompted.

`ssh-keygen` generates a private key file and a public key file in `~/.ssh` (unless you specify an alternate directory for the private key file):

```
$ ls -l ~/.ssh
total 8
-rw-------. 1 guest guest 1743 Apr 13 12:07 id_rsa
-rw-r--r--. 1 guest guest  397 Apr 13 12:07 id_rsa.pub
```

For more information, see the `ssh-keygen(1)` manual page.

### 4.1.4 Enabling Remote System Access Without Requiring a Password

If you want to use OpenSSH utilities to access a remote system without supplying a password each time that you connect, perform these steps:

1. Use `ssh-keygen` to generate a public and private key pair, for example:

```
$ 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>
```

Press `Enter` each time you are prompted to enter a passphrase.

2. Use the `ssh-copy-id` script to append the public key in the local `~/.ssh/id_rsa.pub` file to the `~/.ssh/authorized_keys` file on the remote system, for example:

```
$ ssh-copy-id remote_user@host
remote_user@host's password: remote_password
Now try logging into the machine, with "ssh 'remote_user@host'", and check in:

    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

When prompted, enter your password for the remote system.

The script also changes the permissions of `~/.ssh` and `~/.ssh/authorized_keys` on the remote system to disallow access by your group.

You can now use the OpenSSH utilities to access the remote system without supplying a password. As the script suggests, you should use `ssh` to log into the remote system to verify that the `~/.ssh/authorized_keys` file contains only the keys for the systems from which you expect to connect, for example:

```
$ ssh remote_user@host
Last login: Thu Jun 13 08:33:58 2013 from local_host
```
3. Verify that the permissions on the remote ~/.ssh directory and ~/.ssh/authorized_keys file allow access only by you:

```bash
$ ssh remote_user@host ls -al .ssh
```

```
total 4
drwx------ 2 remote_user group 5 Jun 12 08:33 .
drw-xr-x 9 remote_user group 9 Jun 12 08:32 ..
-rw------- 1 remote_user group 397 Jun 12 08:33 authorized_keys
```

```bash
$ ssh remote_user@host getfacl .ssh
```

```
# file: .ssh
# owner: remote_user
# group: group
user::rwx
group::---
mask::rwx
other::---
```

```bash
$ ssh remote_user@host getfacl .ssh/authorized_keys
```

```
# file: .ssh/authorized_keys
# owner: remote_user
# group: group
user::rw-
group::---
mask::rwx
other::---
```

If necessary, correct the permissions by using the following command:

```bash
$ ssh remote_user@host 'umask 077; /sbin/restorecon .ssh'
$ ssh remote_user@host 'umask 077; /sbin/restorecon .ssh/authorized_keys'
```

If your user names are the same on the client and the server systems, you do not need to specify your remote user name and the @ symbol.

4. If your user names are different on the client and the server systems, create a ~/.ssh/config file with permissions 600 on the remote system that defines your local user name, for example:

```bash
$ ssh remote_user@host echo -e "Host *
User local_user"
```

You should now be able to access the remote system without needing to specify your remote user name by typing the following command:

```bash
$ ssh host ls -l .ssh/config
```

```
-rwx------ 1 remote_user group 37 Jun 12 08:34 .ssh/config
```

For more information, see the ssh-copy-id(1), ssh-keygen(1), and ssh_config(5) manual pages.
4.2 Restricting Access to SSH Connections

The Secure Shell (SSH) allows protected, encrypted communications with other systems. Because SSH is an entry point into the system, disable it if it is not required. Alternatively, you edit the `/etc/ssh/sshd_config` file to restrict its use.

For example, the following setting does not allow `root` to log in by using SSH:

```
PermitRootLogin no
```

You can restrict remote access to certain users and groups by specifying the `AllowUsers`, `AllowGroups`, `DenyUsers`, and `DenyGroups` settings, for example:

```
DenyUsers carol dan
AllowUsers alice bob
```

The `ClientAliveInterval` and `ClientAliveCountMax` settings cause the SSH client to time out automatically after a period of inactivity, for example:

```
# Disconnect client after 300 seconds of inactivity
ClientAliveCountMax 0
ClientAliveInterval 300
```

After making changes to the configuration file, restart the `sshd` service for the changes to take effect.

For more information, see the `sshd_config(5)` manual page.