Working with SSH Servers, Clients, and Keys

Computer and Networking Project
(Information Technology)

Submitted By:
Amritpal Singh
(1411234, D_2-ITA_1)
Gursimar Singh
(1411256, D_2-ITA_1)

Submitted To:
Prof. Lovepreet Kaur
Computer and Networking
System

Department of Information Technology
Guru Nanak Dev Engineering College, Ludhiana 141006

CONTENTS

1 Overview

The most common way of connecting to a remote Linux server is through SSH. SSH stands for Secure Shell and provides a safe and secure way of executing commands, making changes, and configuring services remotely. When you connect through SSH, you log in using an account that exists on the remote server.

2 How SSH Works

When you connect through SSH, you will be dropped into a shell session, which is a text-based interface where you can interact with your server. For the duration of your SSH session, any commands that you type into your local terminal are sent through an encrypted SSH tunnel and executed on your server.
The SSH connection is implemented using a client-server model. This means that for an SSH connection to be established, the remote machine must be running a piece of software called an SSH daemon. This software listens for connections on a specific network port, authenticates connection requests, and spawns the appropriate environment if the user provides the correct credentials.
The user's computer must have an SSH client. This is a piece of software that knows how to communicate using the SSH protocol and can be given information about the remote host to connect to, the username to use, and the credentials that should be passed to authenticate. The client can also specify certain details about the connection type they would like to establish.

3 How SSH Authenticates Users

Clients generally authenticate either using passwords (less secure and not recommended) or SSH keys, which are very secure.
Password logins are encrypted and are easy to understand for new users. However, automated bots and malicious users will often repeatedly try to authenticate to accounts that allow password-based logins, which can lead tosecurity compromises. For this reason, we recommend always setting up SSH- based authentication for most configurations.
SSH keys are a matching set of cryptographic keys which can be used for authentication. Each set contains a public and a private key. The public key can be shared freely without concern, while the private key must be vigilantly guarded and never exposed to anyone.
To authenticate using SSH keys, a user must have an SSH key pair on their local computer. On the remote server, the public key must be copied to a file within the user's home directory at /.ssh/authorized_keys . This file contains a list of public keys, one-per-line, that are authorized to log into this account. When a client connects to the host, wishing to use SSH key authentication, it will inform the server of this intent and will tell the server which public key to use. The server then check its authorized_keys file for the public key, generate a random string and encrypts it using the public key. This encrypted message can only be decrypted with the associated private key. The server will send this encrypted message to the client to test whether they actually have the associated private key.
Upon receipt of this message, the client will decrypt it using the private key and combine the random string that is revealed with a previously negotiated session ID. It then generates an MD5 hash of this value and transmits it back to the server. The server already had the original message

and the session ID, so it can compare an MD5 hash generated by those values and determine that the client must have the private key.
Now that you know how SSH works, we can begin to discuss some examples to demonstrate different ways of working with SSH

4 Generating and Working with SSH Keys

This section will cover how to generate SSH keys on a client machine and distribute the public key to servers where they should be used. This is a good section to start with if you have not previously generated keys due to the increased security that it allows for future connections.

4.1 Generating an SSH Key Pair

Generating a new SSH public and private key pair on your local computer is the first step towards authenticating with a remote server without a password. Unless there is a good reason not to, you should always authenticate using SSH keys.
A number cryptographic algorithms can be used to generate SSH keys, including RSA, DSA, and ECDSA. RSA keys are generally preferred and are the default key type.
To generate an RSA key pair on your local computer, type:

```
ssh-keygen
    Generating public/private rsa key pair.
    Enter file in which to save the key (/home/demo/.ssh/id_rsa):
```

This prompt allows you to choose the location to store your RSA private key. Press ENTER to leave this as the default, which will store them in the.sshhidden directory in your user's home directory. Leaving the default location selected will allow your SSH client to find the keys automatically.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

The next prompt allows you to enter a passphrase of an arbitrary length to secure your private key. By default, you will have to enter any passphrase you set here every time you use the private key, as an additional security measure. Feel free to press ENTER to leave this blank if you do not want a passphrase. Keep in mind though that this will allow anyone who gains control of your private key to login to your servers. If you choose to enter a passphrase, nothing will be displayed as you type. This is a security precaution.

```
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
8c:e9:7c:fa:bf:c4:e5:9c:c9:b8:60:1f:fe:1c:d3:8a root@here
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|       +         |
```

```
|         o S     .       |
|        o     . * +      |
|         o + = O  .      |
|          + = = +        |
|         . . . . Eo+     |
+-----------------------+
```

This procedure has generated an RSA SSH key pair, located in the.sshhidden directory within your user's home directory. These files are:

~/.ssh/id_rsa:

The private key. DO NOT SHARE THIS FILE!

~/.ssh/id_rsa.pub:

The associated public key. This can be shared freely without consequence.

4.2 Removing or Changing the Passphrase on a Private Key

If you have generated a passphrase for your private key and wish to change or remove it, you can do so easily.
Note: To change or remove the passphrase, you must know the original passphrase. If you have lost the passphrase to the key, there is no recourse and you will have to generate a new key pair.
To change or remove the passphrase, simply type:

```
ssh-keygen -p
Enter file in which the key is (/root/.ssh/id_rsa):
```

You can type the location of the key you wish to modify or press ENTER to accept the default value:

```
Enter old passphrase:
```

Enter the old passphrase that you wish to change. You will then be prompted for a new passphrase:

```
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
```

Here, enter your new passphrase or press ENTER to remove the passphrase.

4.3 Displaying the SSH Key Fingerprint

Each SSH key pair share a single cryptographic "fingerprint" which can be used to uniquely identify the keys. This can be useful in a variety of situations.
To find out the fingerprint of an SSH key, type:

```
ssh-keygen -l
    Enter file in which the key is (/root/.ssh/id_rsa):
```

You can press ENTER if that is the correct location of the key, else enter the revised location. You will be given a string which contains the bit-length of the key, the fingerprint, and account and host it was created for, and the algorithm used:

```
4096 8e:c4:82:47:87:c2:26:4b:68:ff:96:1a:39:62:9e:4e  demo@test (RSA)
```

4.4 Copying your Public SSH Key to a Server with SSH-Copy-ID

To copy your public key to a server, allowing you to authenticate without a password, a number of approaches can be taken.
If you currently have password-based SSH access configured to your server, and you have thessh-copy-idutility installed, this is a simple process. Thessh-copy-idtool is included in many Linux distributions' OpenSSH packages, so it very likely may be installed by default. If you have this option, you can easily transfer your public key by typing:

```
ssh-copy-id username@remote_host
```

This will prompt you for the user account's password on the remote system:

```
The authenticity of host '111.111.11.111 (111.111.11.111)' can't
be established. ECDSA key fingerprint is
fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if
you are prompted now it is to install the new keys
demo@111.111.11.111's password:
```

After typing in the password, the contents of your /.ssh/id_rsa.pubkey will be appended to the end of the user account's /.ssh/authorized_keysfile:

```
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'demo@111.111.11.111'"
and check to make sure that only the key(s) you wanted were added.
You can now log into that account without a password:
ssh username@remote_host
```

4.5 Copying your Public SSH Key to a Server Without SSH-Copy-ID

If you do not have thessh-copy-idutility available, but still have password-based SSH access to the remote server, you can copy the contents of your public key in a different way.
You can output the contents of the key and pipe it into thesshcommand. On the remote side, you can ensure that the /.sshdirectory exists, and then append the piped contents into the /.ssh/authorized_keysfile

```
cat ~/.ssh/id_rsa.pub | ssh username@remote_host "mkdir -p ~/.ssh &&
cat >> ~/.ssh/authorized_keys"
```

You will be asked to supply the password for the remote account:

```
The authenticity of host '111.111.11.111 (111.111.11.111)' can't
be established.
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
Are you sure you want to continue connecting (yes/no)? yes
demo@111.111.11.111's password:
```

After entering the password, your key will be copied, allowing you to log in without a password:

```
ssh username@remote_IP_host
```

4.6 Copying your Public SSH Key to a Server Manually

If you do not have password-based SSH access available, you will have to add your public key to the remote server manually.
On your local machine, you can find the contents of your public key file by typing:

```
cat ~/.ssh/id_rsa.pub
```

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCqql6MzstZYh1TmWWv11q5O3pISj2ZFl9H
gH1JLknLLx44+tXfJ7mIrKNxOOwxIxvcBF8PXSYvobFYEZjGIVCEAjrUzLiIxbyC
oxVyle7Q+bqgZ8SeeM8wzytsY+dVGcBxF6N4JS+zVk5eMcV385gG3Y6ON3EG112n6
d+SMXY0OEBIcO6x+PnUSGHrSgpBgX7Ks1r7xqFa7heJLLt2wWwkARptX7udSq05pa
BhcpB0pHtA1Rfz3K2B+ZVIpSDfki9UVKzT8JUmwW6NNzSgxUfQHGwnW7kj4jp4AT0
VZk3ADw497M2G/12N0PPB5CnhHf7ovgy6nL1ikrygTKRFmNZISvAcywB9GVqNAVE+
ZHDSCuURNsAInVzgYo9xgJDW8wUw2o8U77+xiFxgI5QSZX3Iq7YLMgeksaO4rBJEa
54k8m5wEiEE1nUhLuJ0X/vh2xPff6SQ1BL/zkOhvJCACK6Vb15mDOeCSq54Cr7kvS4
6itMosi/uS66+PujOO+xt/2FWYepz6ZlN70bRly57Q06J+ZJoc9FfBCbCyYH7U/AS
smY095ywPsBo1XQ9PqhnN1/YOorJ068foQDNVpm146mUpILVxmq41Cj55YKHEazXG
sdBIbXWhcrRf4G2fJLRcGUr9q8/lERo9oxRm5JFX6TCmj6kmiFqv+Ow9gI0x8Gva
Q==demo@test
```

You can copy this value, and manually paste it into the appropriate location on the remote server. You will have to log into the remote server through other means (like the DigitalOcean web console). On the remote server, create the /.sshdirectory if it does not already exist:

```
mkdir -p ~/.ssh
```

Afterwards, you can create or append the /.ssh/authorized_keysfile by typing:

```
echo public_key\_string >> ~/.ssh/authorized_keys
```

You should now be able to log into the remote server without a password.

5 Basic Connection Instructions

The following section will cover some of the basics about how to connect to a server with SSH.

5.1 Connecting to a Remote Server

To connect to a remote server and open a shell session there, you can use thesshcommand.
The simplest form assumes that your username on your local machine is the same as that on the remote server. If this is true, you can connect using:

```
ssh remote_host
```

If your username is different on the remoter server, you need to pass the remote user's name like this:

```
ssh username@remote_host
```

Your first time connecting to a new host, you will see a message that looks like this:

```
The authenticity of host '111.111.11.111 (111.111.11.111)' can't be established
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
Are you sure you want to continue connecting (yes/no)? yes
```

Type "yes" to accept the authenticity of the remote host. If you are using password authentication, you will be prompted for the password for the remote account here. If you are using SSH keys, you will be prompted for your private key's passphrase if one is set, otherwise you will be logged in automatically.

5.2 Running a Single Command on a Remote Server

To run a single command on a remote server instead of spawning a shell session, you can add the command after the connection information, like this:

```
ssh username@remote_host command_to_run
```

This will connect to the remote host, authenticate with your credentials, and execute the command you specified. The connection will immediately close afterwards.

5.3 Logging into a Server with a Different Port

By default the SSH daemon on a server runs on port 22. Your SSH client will assume that this is the case when trying to connect. If your SSH server is listening on a non-standard port (this is demonstrated in a later section), you will have to specify the new port number when connecting with your client. You can do this by specifying the port number with the-poption:

```
ssh -p port_num username@remote_host
```

To avoid having to do this every time you log into your remote server, you can create or edit a configuration file in the /.sshdirectory within the home directory of your local computer.
Edit or create the file now by typing:

```
nano ~/.ssh/config
```

In here, you can set host-specific configuration options. To specify your new port, use a format like this:

```
Host remote_alias
    HostName remote_host
    Port port_num
```

This will allow you to log in without specifying the specific port number on the command line.

BIBLIOGRAPHY

[1] https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys

[2] LaTeX Beginner's Guide By Stefan Kottwitz [Pact Publishing]

[3] My Blog, http://www.amritpals.com

[4] My Github Profile, http://github.com/amrit3701