Key based login

Step 1 — Creating SSH Keys

The first step to configure SSH key authentication to your server is to generate an SSH key pair on your local computer.

To do this, we can use a special utility called `ssh-keygen`, which is included with the standard OpenSSH suite of tools. By default, this will create a 3072 bit RSA key pair.

On your local computer, generate a SSH key pair by typing:

==========

MY TERMINAL

lab05-pc04@lab05pc04-H310M-S2-2-0:~$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/home/lab05-pc04/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/lab05-pc04/.ssh/id_rsa

Your public key has been saved in /home/lab05-pc04/.ssh/id_rsa.pub

The key fingerprint is:

SHA256:e4WH/+zJ9MTepemtVsBWdgFpWF56fsy67QH+/YhYOHY lab05-pc04@lab05pc04-H310M-S2-2-0

The key's randomart image is:

```
+---[RSA 3072]----+
|           oooo.|
|          ..oo +|
|           .+ +.|
|          o  *o |
|        S o oo o+|
|         . =. .oo|
|        . = E.oo+|
|         o = *o%=|
|          . .+&+@|
+----[SHA256]-----+
```

```
================================================================
```

The utility will prompt you to select a location for the keys that will be generated. By default, the keys will be stored in the `~/.ssh` directory within your user's home directory. The private key will be called `id_rsa` and the associated public key will be called `id_rsa.pub`.

Usually, it is best to stick with the default location at this stage. Doing so will allow your SSH client to automatically find your SSH keys when attempting to authenticate. If you would like to choose a non-standard path, type that in now, otherwise, press `ENTER` to accept the default.

If you had previously generated an SSH key pair, you may see a prompt that looks like this:

```
Output
/home/username/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

If you choose to overwrite the key on disk, you will **not** be able to authenticate using the previous key anymore. Be very careful when selecting yes, as this is a destructive process that cannot be reversed.

```
Output
Created directory '/home/username/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Next, you will be prompted to enter a passphrase for the key. This is an optional passphrase that can be used to encrypt the private key file on disk.

You may be wondering what advantages an SSH key provides if you still need to enter a passphrase. Some of the advantages are:

- The private SSH key (the part that can be passphrase protected), is never exposed on the network. The passphrase is only used to decrypt the key on the local machine. This means that network-based brute forcing will not be possible against the passphrase.
- The private key is kept within a restricted directory. The SSH client will not recognize private keys that are not kept in restricted directories. The key itself must also have restricted permissions (read and write only available for the owner). This means that other users on the system cannot snoop.
- Any attacker hoping to crack the private SSH key passphrase must already have access to the system. This means that they will already have access to your user account or the root account. If you are in this position, the passphrase can prevent the attacker from immediately logging into your other servers. This will hopefully give you time to create and implement a new SSH key pair and remove access from the compromised key.

Since the private key is never exposed to the network and is protected through file permissions, this file should never be accessible to anyone other than you (and the **root** user). The passphrase serves as an additional layer of protection in case these conditions are compromised.

A passphrase is an optional addition. If you enter one, you will have to provide it every time you use this key (unless you are running SSH agent software that stores the decrypted key). We recommend using a passphrase, but if you do not want to set a passphrase, you can press `ENTER` to bypass this prompt.

```
Output
Your identification has been saved in /home/username/.ssh/id_rsa.
Your public key has been saved in /home/username/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:CAjsV9M/tt5skazroTc1ZRGCBz+kGtYUIPhRvvZJYBs username@hostname
The key's randomart image is:
+---[RSA 3072]----+
|o   ..oo.++o ..  |
| o o +o.o.+...   |
|. . + oE.o.o  .  |
| . . oo.B+  .o   |
|  .   .=S.+ +    |
|      . o..*     |
|        .+= o    |
|        .=.+     |
|       .oo+      |
+----[SHA256]-----+
```

You now have a public and private key that you can use to authenticate. The next step is to place the public key on your server so that you can use SSH key authentication to log in.

==========

Step 2 — Copying an SSH Public Key to Your Server

There are multiple ways to upload your public key to your remote SSH server. The method you use depends largely on the tools you have available and the details of your current configuration.

The following methods all yield the same end result. The simplest, most automated method is described first, and the ones that follow it each require additional manual steps. You should follow these only if you are unable to use the preceding methods.

Copying Your Public Key Using `ssh-copy-id`

The simplest way to copy your public key to an existing server is to use a utility called `ssh-copy-id`. Because of its simplicity, this method is recommended if available.

The `ssh-copy-id` tool is included in the OpenSSH packages in many distributions, so you may already have it available on your local system. For this method to work, you must currently have password-based SSH access to your server.

To use the utility, you need to specify the remote host that you would like to connect to, and the user account that you have password-based SSH access to. This is the account where your public SSH key will be copied.

==========l

MY TERMINAL

lab05-pc04@lab05pc04-H310M-S2-2-0:~$ ssh-copy-id umamahesh@172.16.150.82
/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
umamahesh@172.16.150.82's password:
/home/umamahesh/.bashrc: line 29: alias: a: not found

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'umamahesh@172.16.150.82'"
and check to make sure that only the key(s) you wanted were added.

==========

The syntax is:

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

You may see a message like this:

```
Output
The authenticity of host '203.0.113.1 (203.0.113.1)' 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
```

This means that your local computer does not recognize the remote host. This will happen the first time you connect to a new host. Type `yes` and press `ENTER` to continue.

Next, the utility will scan your local account for the `id_rsa.pub` key that we created earlier. When it finds the key, it will prompt you for the password of the remote user's account:

```
Output
/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
username@203.0.113.1's password:
```

Type in the password (your typing will not be displayed for security purposes) and press `ENTER`. The utility will connect to the account on the remote host using the password you provided. It will then copy the contents of your `~/.ssh/id_rsa.pub` key into a file in the remote account's home `~/.ssh` directory called `authorized_keys`.

You will see output that looks like this:

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

Now try logging into the machine, with:   "ssh 'username@203.0.113.1'"
and check to make sure that only the key(s) you wanted were added.
```

At this point, your `id_rsa.pub` key has been uploaded to the remote account. You can continue onto the next section.

Copying Your Public Key Using SSH

If you do not have `ssh-copy-id` available, but you have password-based SSH access to an account on your server, you can upload your keys using a conventional SSH method.

We can do this by outputting the content of our public SSH key on our local computer and piping it through an SSH connection to the remote server. On the other side, we can make sure that the `~/.ssh` directory exists under the account we are using and then output the content we piped over into a file called `authorized_keys` within this directory.

We will use the `>>` redirect symbol to append the content instead of overwriting it. This will let us add keys without destroying previously added keys.

The full command will look like this:

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

You may see a message like this:

```
Output
The authenticity of host '203.0.113.1 (203.0.113.1)' 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
```

This means that your local computer does not recognize the remote host. This will happen the first time you connect to a new host. Type `yes` and press `ENTER` to continue.

Afterwards, you will be prompted with the password of the account you are attempting to connect to:

```
Output
username@203.0.113.1's password:
```

After entering your password, the content of your `id_rsa.pub` key will be copied to the end of the `authorized_keys` file of the remote user's account. Continue to the next section if this was successful.

Copying Your Public Key Manually

If you do not have password-based SSH access to your server available, you will have to do the above process manually.

The content of your `id_rsa.pub` file will have to be added to a file at `~/.ssh/authorized_keys` on your remote machine somehow.

To display the content of your `id_rsa.pub` key, type this into your local computer:

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

You will see the key's content, which may look something like this:

~/.ssh/id_rsa.pub

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCqql6MzstZYh1TmWWv11q5O3pISj2ZFl9HgH1JLknLLx44+tXf
J7mIrKNxOOwxIxvcBF8PXSYvobFYEZjGIVCEAjrUzLiIxbyCoxVyle7Q+bqgZ8SeeM8wzytsY+dVGcBx
F6N4JS+zVk5eMcV385gG3Y6ON3EG112n6d+SMXY0OEBIcO6x+PnUSGHrSgpBgX7Ks1r7xqFa7heJLLt2
wWwkARptX7udSq05paBhcpB0pHtA1Rfz3K2B+ZVIpSDfki9UVKzT8JUmwW6NNzSgxUfQHGwnW7kj4jp4
AT0VZk3ADw497M2G/
12N0PPB5CnhHf7ovgy6nL1ikrygTKRFmNZISvAcywB9GVqNAVE+ZHDSCuURNsAInVzgYo9xgJDW8wUw2
o8U77+xiFxgI5QSZX3Iq7YLMgeksaO4rBJEa54k8m5wEiEE1nUhLuJ0X/vh2xPff6SQ1BL/
zkOhvJCACK6Vb15mDOeCSq54Cr7kvS46itMosi/uS66+PujOO+xt/
2FWYepz6ZlN70bRly57Q06J+ZJoc9FfBCbCyYH7U/ASsmY095ywPsBo1XQ9PqhnN1/
YOorJ068foQDNVpm146mUpILVxmq41Cj55YKHEazXGsdBIbXWhcrRf4G2fJLRcGUr9q8/
lERo9oxRm5JFX6TCmj6kmiFqv+Ow9gI0x8GvaQ== username@hostname
```

Access your remote host using whatever method you have available. This may be a web-based console provided by your infrastructure provider.

Note: if you're using a DigitalOcean Droplet, please refer to our Recovery Console documentation in the DigitalOcean product docs.

Once you have access to your account on the remote server, you should make sure the `~/.ssh` directory is created. This command will create the directory if necessary, or do nothing if it already exists:

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

Now, you can create or modify the `authorized_keys` file within this directory. You can add the contents of your `id_rsa.pub` file to the end of the `authorized_keys` file, creating it if necessary, using this:

```
echo public_key_string >> ~/.ssh/authorized_keys
```

In the above command, substitute the with the output from the `cat ~/.ssh/id_rsa.pub` command that you executed on your local system. It should start with `ssh-rsa AAAA...` or similar.

If this works, you can move on to test your new key-based SSH authentication.

Step 3 — Authenticating to Your Server Using SSH Keys

If you have successfully completed one of the procedures above, you should be able to log into the remote host *without* the remote account's password.

The process is mostly the same:

```
ssh username@remote_host
```

If this is your first time connecting to this host (if you used the last method above), you may see something like this:

```
Output
The authenticity of host '203.0.113.1 (203.0.113.1)' 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
```

This means that your local computer does not recognize the remote host. Type `yes` and then press `ENTER` to continue.

If you did not supply a passphrase for your private key, you will be logged in immediately. If you supplied a passphrase for the private key when you created the key, you will be required to enter it now. Afterwards, a new shell session will be created for you with the account on the remote system.

If successful, continue on to find out how to lock down the server.

Step 4 — Disabling Password Authentication on your Server

If you were able to login to your account using SSH without a password, you have successfully configured SSH key-based authentication to your account. However, your password-based authentication mechanism is still active, meaning that your server is still exposed to brute-force attacks.

Before completing the steps in this section, make sure that you either have SSH key-based authentication configured for the **root** account on this server, or preferably, that you have SSH key-based authentication configured for an account on this server with `sudo` access. This step will lock down password-based logins, so ensuring that you will still be able to get administrative access is essential.

Once the above conditions are true, log into your remote server with SSH keys, either as **root** or with an account with `sudo` privileges. Open the SSH daemon's configuration file:

```
sudo nano /etc/ssh/sshd_config
```

Inside the file, search for a directive called `PasswordAuthentication`. This may be commented out. Uncomment the line by removing any # at the beginning of the line, and set the value to `no`. This will disable your ability to log in through SSH using account passwords:

/etc/ssh/sshd_config

```
PasswordAuthentication no
```

Save and close the file when you are finished. To actually implement the changes we just made, you must restart the service.

On most Linux distributions, you can issue the following command to do that:

```
sudo systemctl restart ssh
```

After completing this step, you've successfully transitioned your SSH daemon to only respond to SSH keys.

Conclusion

You should now have SSH key-based authentication configured and running on your server, allowing you to sign in without providing an account password.