Linux Academy

Hands-on Lab

Generate, Exchange, and Use SSH Keys for Authentication

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- The IP address of your server is located on the Hands-on Lab page
- Username: linuxacademy
- Password: 123456

Related Courses

LPIC-2 Linux Engineer Exam 202

Related Videos

Using SSH Keys for Authentication

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

In this lab, we generate an SSH key for authentication into a server without the use of a password. We use two CentOS 6 servers to generate, exchange, and use the SSH keys to authorize access.

Generate the SSH Key

Log in to the first server using the credentials provided on the Hands-on Lab page.

We want to generate a key on the server that is making the connections – it is not a two-way process between the servers.

Generate the key:

```
[linuxacademy@server1] ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/linuxacademy/.ssh/id_rsa):
Created directory '/home/linuxacademy/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/linuxacademy/.ssh/id_rsa.
Your public key has been saved in /home/linuxacademy/.ssh/id_rsa.pub.
The key fingerprint is:
66:d5:09:52:f3:ec:f4:e3:58 linuxacademy@ip-10-0-1-240
The key's randomart image is:
+--[ RSA 2048]----+
+-----------------+
```

We can define a name for the key we want to create or use the default, `id_rsa`. For this lab, we are using the default name. Additionally, we are *not* adding a passphrase for this key.

Should we list the contents of the `.ssh` directory, we can see our added private and public keys:

```
[linuxacademy@server1] ls .ssh
id_rsa  id_rsa.pub
```

Copy Key to Remote System (ssh-copy-id)

The following section is not on the LPIC exam.

To copy the public key to a remote system we can use the `ssh-copy-id` command. We want to copy our key to our second server.

```
[linuxacademy@server1] ssh-copy-id linuxacademy@SERVER2PRIVATEIP
```

We should now SSH into the second server from you workstation and ensure nothing extra was added to

the `.ssh/authorized_keys` file:

```
[linuxacademy@server2] cat .ssh/authorized_keys
ssh-rsa
z1Q4VxtheUmUFbOUdOPmGhjttgSiFQZoEvxR6O4OtrRtoHFlxfZph8xpBRud3NCQnr//
u5U3X1XfQ9Dkm5V++NAnQ7qcjkFVdSxf+linuxacademy@ip-10-0-1-240
```

We can now try to SSH into the server from the first server:

```
[linuxacademy@server1] ssh linuxacademy@SERVER2PRIVATEIP
```

As we can see, we accessed the server without a password or passphrase. `exit` the server.

Copy Key to Remote System (Manual)

Since `ssh-copy-id` is not included in the LPIC syllabus, we instead need to know how to copy over the SSH key manually.

For the first server, `cat` out the public key and pipe it through `ssh` and append the key to the `.ssh/authorized_keys` file:

```
[linuxacademy@server1] cat .ssh/id_rsa.pub | ssh@linuxacademy@
SERVER2PRIVATEIP 'cat >> .ssh/authorized_keys'
```

In this instance, we are not prompted to input a password because the `.ssh/authorized_keys` file already contains the public key for the first server.

Should we return to the second server and view the `.ssh/authorized_keys` file, we can see the public key has been added twice:

```
[linuxacademy@server2] cat .ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAzuCIz1Q4VxtheUmUFbOUdOPmGhjttgSiFQZoE==
linuxacademy@ip-10-0-1-240
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAzuCIz1Q4VxtheUmUFbOUdOPmGhjttgSiFQZoE==
linuxacademy@ip-10-0-1-240
```

Open the `.ssh/authorized_keys` file in your preferred text editor and remove the duplicate entry.

On the first server, confirm that we can still log in:

```
[linuxacademy@server1] ssh linuxacademy@SERVER2PRIVATEKEY
```

Once confirmed, `exit`.

Generate Additional Keys with Passphrases

From the second server, remove any keys from the `.ssh/authorized_keys` file. Return to the first server.

Using `ssh-keygen`, overwrite the `id_rsa` key, this time adding a passphrase when prompted.

```
[linuxacademy@server1] ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/linuxacademy/.ssh/id_rsa):
/home/linuxacademy/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/linuxacademy/.ssh/id_rsa.
Your public key has been saved in /home/linuxacademy/.ssh/id_rsa.pub.
The key fingerprint is:
03:0b:cfed:f7:15:51:1d linuxacademy@ip-10-0-1-240
The key's randomart image is:
+--[ RSA 2048]----+
+-----------------+
```

Copy the key to the second server:

```
[linuxacademy@server1] ssh-copy-id linuxacademy@SERVER2PRIVATEIP
```

Now, check the second server to see that the key has, again, been added to the `.ssh/authorized_keys` file:

```
[linuxacademy@server2] cat .ssh/authorized_keys
ssh-rsa
yc2EAAAABIwAAAQEAoq7vdkZULIrLz2or56nWV7KKr3hN4OZLmTIRIgDX3NHDRyzdPxwg/
IWXWp9Jg== linuxacademy@ip-10-0-1-240
```

If we compare with earlier keys, we can see that this is a different key than before.

Return to the first server. SSH into the second:

```
[linuxacademy@server1] ssh linuxacademy@SEVER2PRIVATEIP
```

This time, we are prompted to input a passphrase for our private key (versus a user password).

While this might seem like it defeats the purpose of adding an SSH key, this can work as two-factor

authentication, to ensure that whoever is attempting access to the server should have access.

`exit` server two, so we are back to the first server.

However, as users there is a way to get around this, but inputting the password in only once. Meaning, the key is associated with the SSH session so that the passphrase only has to be input once per session regardless of how many servers are SSHed into.

To do this, we use `ssh-agent`:

```
[linuxacademy@server1] ssh-agent /bin/bash
[linuxacademy@server1] ssh-add
```

Now, log in to the second server:

```
[linuxacademy@server1] ssh linuxacademy@SERVER2PRIVATEIP
```

We are not prompted for the passphrase. However, should we `exit` the Bash session, we will have to input a passphrase again.

This lab is now complete!