15.10.1 Public Key Authentication

Click one of the buttons to take you to that part of the video.

Public Key Authentication 0:00-0:32

In addition to authenticating to an SSH server using a standard username and password combination, you can also reconfigure your sshd daemon such that it will allow authentication using either an RSA or DSA public key.

In which case, when you connect to the server system using your SSH client, you will not be prompted for a username and password. Instead, you will submit your public key. And if it's the right key, you are authenticated to the system.

Public Key Authentication 0:33-3:41

In order for this to work, we have to pre-provision the server with the public key of the user on the client system.

What we will need to do is take the public key from the user over here on the client system and add it manually to the authorized_keys file located in the /.ssh hidden directory of the user's /home directory on the server system that you're going to authenticate as.

For example, let's suppose that I log in locally to this system as the ksanders user, but I want to authenticate to the remote system where the sshd daemon is running as the rtracy user. In this case, I'm going to have to come over here on the client system and generate a private/public key pair for the ksanders user.

Then I'm going to need to manually transfer the public key from the ksanders user over here to the authorized_keys file located in the /.ssh hidden directory in the /home directory of the rtracy user, because that's who I'm going to try to authenticate to the remote system as.

Remember, because we're dealing with a private/public key pair, only the public key is transferred from the client system to the server system. The private key always stays over here on the client system itself because, remember, any data encrypted with the public key can be decrypted only by the private key.

When you configure the SSH server over here to use public key authentication, the SSH client running over here on the client system tells the SSH server which public key it is that should be used for authentication when the SSH session is initially established. The SSH server will then check and see if it has that client's public key.

If it does, it will generate a random number, and it will encrypt it with that public key. It will then send that encrypted number back over here to the client system. Remember, it's got the private key, so it is able to decrypt that random number because it was encrypted over here on the server system using this system's public key.

After decrypting the random number, the client system over here is going to run the MD5 checksum on that number, and it's going to send the value of that checksum back over to the SSH system over here. This system is going to do the same thing.

It's going to calculate an MD5 checksum of the number that it originally sent down here. Then it's going to compare the number it got over here with the value of the MD5 checksum that was sent from the client system.

The key point here is that if this checksum matches this checksum, then we know that the public/private key pair is valid; and, therefore, the user is automatically logged in over here on the SSH server.

Client Key Pair 3:42-5:17

In order to configure public key authentication, the first thing you need to do is create your private/public key pair on the client system so you can send the public key that results over to the SSH server. This is done using the ssh-keygen command.

You can either create an RSA key pair or a DSA key pair, either one is fine. In fact, when I do this, I usually create both just so I have them.

Whichever command you use, two keys will be created as a result. You have a private key and a public key. This is the RSA private key/public key pair, and this is the DSA private/public key pair.

If you use RSA, then the private key will be saved in the /.ssh hidden directory in your /home directory. It's named id_rsa, while your public key will be in the same directory but will have an extension of .pub. The same holds true if you create a DSA key pair.

Regardless of which encryption mechanism you choose to use, RSA or DSA, you will be prompted during the key generation process to create a passphrase for that key. There are two important things you need to be aware of with respect to this passphrase.

First of all, you should use a passphrase, because you will be given the option to not use one if you don't want to. That would be unwise because if you don't, then anyone who managed to get a copy of your key files could authenticate to the SSH server without being required to enter the passphrase.

Assigning a passphrase to the key renders the key useless if somebody doesn't know what that passphrase is. The second thing you need to do is remember what that passphrase is, because you're going to need it.

Public Key Copy 5:18-7:29

Once done, the next thing you need to do is copy the public key that you just created on the client over to the SSH server. A very easy way to do this--and a fairly secure way to do this-- is to use the scp command, secure copy. The syntax is shown here.

In this case, I am copying my RSA public key I generated from my /home directory to the fs5.corpnet.com server over here. I want to put that key in the rtracy user account. I'm going to just name it keyfile.

This will create a new file named keyfile in the rtracy user account over here, and that keyfile file will actually contain the id_rsa.pub--the public key. At this point, we've got the key file over here on the SSH server, but it's saved in a file named keyfile in the destination user's /home directory. That won't work.

What we have to do is take the contents of that file and append it to the end of the authorized_keys file in that user's /home directory in the /.ssh hidden subdirectory.

An easy way to do this is to just log in using a standard SSH session, where we log in with a username and a password, to connect to the SSH server as the user in question. And then enter this command right here: 'keyfile >>' and then the name of the authorized_key file, in this case, in the '/home directory /.ssh/authorized_keys.'

It's very important when you do this to make sure that you use two greater-than signs, not one. Because if you use one, then the key file that we copied over here from the client system will overwrite the authorized_keys file. And if there were already keys in that file that you needed to use, they're going to get wiped out.

If you use two greater-than signs, then we're going to take the contents of the key file and add it to the end of the authorized_keys file, thereby preserving any public keys that might already be in that file.

Authentication 7:30-8:09

With the public key added to the authorized_keys file, we can now test the configuration to see if it works. If we were already logged in to the remote server with a standard SSH session, we would exit out of it, and then try to establish a new SSH session.

If everything is working correctly, you should be prompted to supply the key file's passphrase instead of the standard username and password, as shown here.

Once you enter the passphrase, you'll be authenticated to the SSH server. Then if everything matches up between the public key on the server and the public key on the client, then no password or username will be required to establish the SSH session.

---

ssh-agent 8:10-9:14

If you want to, you can use the ssh-agent command to eliminate the need to enter that passphrase every time you establish an SSH connection with the remote server. The first thing you need to do is, on the client system, run ssh-agent bash at the shell prompt, and then run ssh-add, followed by the private key that is the companion of the public key that you sent over to the SSH server.

If you sent an RSA public key over to the server, then you'd want to specify your RSA private key located in the /.ssh hidden directory in your /home directory. Or, if you sent a DSA public key over to the SSH server, then you'd want to specify the filename of the DSA public key. You'll then be prompted to specify the key file's passphrase.

When you do that, SSH agent is going to store it in memory, and then it will listen for SSH requests. And when one is created, it will automatically provide the passphrase that you specified for that key.

Summary 9:15-9:19

That's it for this lesson. In this lesson we reviewed how you can use public keys to authenticate an SSH connection.