15.7.2 Configure SSH Port Tunneling

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

Configure SSH Port Tunneling 0:00-0:21

In this demonstration I'm going to show you how to tunnel unencrypted network traffic through an encrypted SSH connection. In order to do this, the system that's running the SSHD daemon has to be configured to allow TCP forwarding. This Linux system here is the one where the SSHD daemon will be run.

SSH Tunnel 0:22-2:13

Before I start it up, I'm going to use the vi editor to open the '/etc/ssh/sshd_config' file and we need to locate the AllowTcpForwarding parameter. Notice it is currently commented out. In order for SSH port tunneling to work, we have to remove this pound sign to turn this option on. I'll go ahead and 'exit' out of our editor.

Let's use the 'systemctl' command, get the 'status' of our 'sshd' daemon. It should not be running. It isn't. Let's go ahead and start it. Let's check the 'status' again, and it is now running.

Now that that's done, we can then go and create an SSH tunnel from our SSH client machine that points to this system here. With our SSHD daemon configured to allow TCP forwarding, what we can do now is use the SSH client on this computer to create an SSH tunnel between the client system and the server system. Let's take a look at how you do this.

It's pretty easy. All you have to do is use the 'ssh' command to load the SSH client just as we would normally, but this time we're going to do something a little bit different. We're going to use the '-X' option, and then we use the '-l' option.

Specify the username on the remote system that we want authenticate to that system as; in this case, we're going to authenticate as the 'rtracy' user. Then we have to specify the name of the host that we want to connect to: 'fs5.corpnet.com'. Press Enter. I do have to authenticate as the rtracy user on the other system.

Example 2:14-4:23

Okay, so we have successfully established an SSH tunnel now between this system, the openSUSE system, and the fs5 system. I'm logged in to fs5 as the rtracy user. Because we used the -X option, we're now able to tunnel X-server traffic from fs5 and display it over here on the openSUSE system.

To see how this works, I can actually type the command to run a graphical application on the remote system and have that application's window displayed here locally on my openSUSE system.

I'm going to run the 'gedit' command to load the gedit editor and there we go. There's my gedit application. It's not actually running locally. It's running on the fs5 system, the graphical display is being piped through my SSH tunnel, and it's being displayed here on this system.

So all the processing is actually happening on fs5 and just the display is being shown here. Because we're using an SSH tunnel, all the data that I pass back and forth is encrypted.

I added some text to the document. I'll hit Save. You'll notice because I'm running this remotely, the application itself is running on the fs5 system over here. I'm actually saving it on the remote system.

It's not being saved on the local hard drive. It's being saved on the remote hard drive. Hit Save. We can go ahead and exit out at this point.

If I do an 'ls' command, we see the document I just created in the rtracy user's /home directory over on the fs5 system. Because we used an SSH tunnel, all the words that I typed in this document were sent from this system over to the fs5 system in encrypted format.

Summary 4:24-4:40

That's it for this demonstration. This demo we talked about tunneling unencrypted network traffic through an encrypted an SSH connection. Then we looked at one specific example where we tunneled X-server traffic through an SSH connection so we could run X applications locally, with all the traffic being sent in encrypted format between the two systems.
