15.7.1 SSH Port Tunneling

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

SSH Port Tunneling 0:00-1:21

One of the key security issues that you have to deal with as a system administrator, is the fact that there are many, commonly used network protocols that transfer information across the network as clear text. As you can imagine, this is not good.

That means anybody sniffing the network medium can capture transmissions and they can gain a wealth of information that you don't want them to have. Good examples of this are the POP3 and IMAP daemons.

Now, in order for your Linux MTA (Mail Transfer Agent) to download email messages to network client systems, say running Thunderbird, then you first have to enable either the POP3 or the IMAP daemon using xinetd.

Now, once this is done then users can use their email client to connect to the MTA and download their email using the appropriate protocol. The problem here, however, as we said, is the fact that both of these daemons transfer data as clear text by default. This means that the usernames that the end user configures to authenticate to the MTA, are sent as clear text.

In addition, all the contents of their email messages as they're downloaded, are sent as clear text. This means anyone with a sniffer and a little bit of know-how, can capture those packets and view the contents of these transmissions. Not a good thing.

SSH Tunneling 1:22-2:01

The good news, however, is that you can use SSH to encrypt network traffic that is normally transmitted as clear text. And this is done by tunneling it through an SSH connection.

This tunnel protects whatever is inside of it. So in this case, we've established an SSH tunnel between the client, the MUA (Mail User Agent), and the mail server, the MTA, and then, within this tunnel, we can safely transmit data back and forth between the mail server and the mail client. So, when I log into my mail server using my email client on my workstation, my username and password are sent through this tunnel.

Sniffing Attack 2:02-4:33

When I download messages from the MTA to my email client, they're sent through this tunnel. And, even if we had an attacker right here who is sniffing packets, trying to get my email username and password or to see what attachments I'm sending in my emails, all they're going to see is junk because the traffic is being encrypted with SSH.

So, can they get a copy of the data? Yep. Can they read it? Nope. It's encrypted.

SSH tunneling is great in situations where the protocol that we're concerned about doesn't natively support encryption. For example, we are going to use an X server to display a desktop over here on an X client system.

Normally this would be a problem because the transmissions between the X server and the X client would be sent clear text. That means a sniffer over here in the middle, would be able to capture everything that came across on my desktop. That could potentially allow them access to any proprietary or sensitive information that I displayed, say in a document or in an email on my desktop. That is not a good thing.

Once again, using openSSH, we can establish an SSH tunnel between the client and the server system. Then we tunnel all of that X server traffic between the X server and the X client through that tunnel. Again, could somebody sniff it? Yep. Could they read it? Nope. All they'll see is gibberish. Now, in order to configure this tunnel, we have to perform some configuration tasks on both ends.

First of all, on the client end, there are two different things we can do to set up the client end of the tunnel. One is to use the -x option when we run the ssh command at the shell prompt, the client command. Another option would be to go into the /etc/ssh/ssh_config file and then set ForwardX11 to a value of yes in that file and save the changes.

Then you need to set up the server end of the tunnel. To do this, you edit the /etc/ssh/sshd_config file. Notice it's not the client configuration file but the server configuration file and then set the X forwarding parameter in this file to a value of yes.

Once done, the X traffic between the X server and the X client system will be encrypted. If an attacker were to sniff the communications, they would not be able to decipher it.

Summary 4:34-4:40

That's it for this lesson. In this lesson, we reviewed how you can tunnel unencrypted network traffic through an SSH tunnel to protect it from being read if it were to be captured by an attacker.

Copyright © 2022 TestOut Corporation All rights reserved.