15.7.3 SSH Port Tunneling Facts

Secure Shell (SSH) port forwarding and tunneling encrypts data from non-secure protocols and then sends the data over a network connection. Non-secure protocols, such as email and X server traffic, can be tunneled through SSH.

This lesson covers the following topics:

- SSH Port tunneling process
- Local and remote port forwarding

SSH Port Tunneling Process

The SSH port tunneling process works as follows:

1. The client sends the non-secure protocol information to the port on the server running the SSH daemon.
2. The SSH daemon intercepts all traffic sent to that port, encrypts it, and sends it to the SSH client.
3. The SSH client receives the encrypted traffic, decrypts it, and forwards it to the default port for the client.
4. The client receives the data on its usual port.

The **/etc/ssh/sshd_config** file configures the SSH daemon on the server. Commonly used options for configuring an SSH tunnel include:

- **AllowTcpForwarding** allows TCP traffic to be sent from the SSH daemon when set to **yes**.
- **ForwardX11** specifies that clients to which requests are forwarded are regarded as untrusted, and have restricted access to certain GUI features.
- **ForwardX11Trusted** specifies that clients to which requests are forwarded are regarded as trusted, and have unrestricted access to all GUI features.
- **X11Forwarding** is used on some distributions instead of **ForwardX11Trusted**.
- **VNC** allows any computer to act as a graphical terminal server that supports multiple desktops and multiple users.

Use the following commands to create an SSH port tunnel:

Command	Function	Example
ssh	Sets up an SSH tunnel from the client to the server. Options include:	**ssh -f -N -L 2345:mail.corpnet.com:110 userbob@mail.corpnet.com** sets up an SSH

	• **-f** runs SSH in the background after the password prompt. • **-N** ensures that SSH does not execute a remote command. • **-L** specifies the port numbers and server address. • **-g** overrides configuration file settings and creates a tunnel (if needed). • **tunnel***port* specifies the SSH port for the encrypted data. Only the root user can set the SSH port to a privileged port (e.g., port 1024 or lower). • **server** specifies the server running the SSH daemon. • *port* specifies the default port for a non-secure protocol.	port tunnel for POP3 mail traffic over port 2345.
ssh -X	Sets up an SSH tunnel from the client to the server for X server traffic. Options include: • **-l** specifies the username of the user account on the remote system. • **server** specifies the SSH server address.	**ssh -X -l mtrance hn3.corpnet.com** sets up an SSH port tunnel for X server traffic.

Local and Remote Port Forwarding

You can redirect the console using a local or a remote SSH connection. The following table describes each of these SSH tunneling methods.

Command	Function
Local Port Forwarding	Uses a local SSH connection to create an encrypted tunnel to a remote machine. With local port forwarding, you connect to a destination server via an SSH server. Be aware of the following: • You enter your password to authenticate to your machine. • To establish the connection you enter **ssh -L** ***port:hostname:port*** **localhost** in a terminal window on your client. For example to connect corpnet.com port 80 to port 8080 on your client machine, enter ssh -L 8080:corpnet.com:80 localhost. • To view the display on compnet.com, you browse to http://localhost 8080. • To close the connection type **exit** in your client terminal window.

Remote Port Forwarding	Uses a remote SSH connection to create an encrypted tunnel from a remote machine to your SSH client. Remote port forwarding is the opposite of local port forwarding. Be aware of the following when using remote port forwarding. • Modify the **/etc/ssh/sshd_config** file to include the a gateway entry at the end of the file. Enter: • ◦ **GatewayPorts yes** allows anyone to connect to the forwarded port. ◦ **Gateway Ports no** prevents access from outside the server computer. ◦ **GatewayPorts clientspecified** allows you to specify an IP address from which connections are supported. • Restart the SSH daemon by using the command **sudo systemctl restart sshd**. This requires that you have ssh access to the remote machine. • Enter the command: **ssh -R** ***port*****:localhost:*port username@hostname***. Enter the username you have access to on the remote machine and the ports your want to use for the connection. • You will have to authenticate to the username on the remote computer.

You can also configure a remote X client without encryption by performing the following:

1. On the client system, enter b>xhost +*server_hostname*. This tells the client to accept connections from the remote X server.
2. On the server system, enter **DISPLAY=*client_hostname*:0.0** and then export the DISPLAY environment variable. This tells the server to send its X display output over the network to the remote client.
3. On the client system, use the SSH client to access the shell prompt on the server.
4. From within the SSH session, run the graphical application you want displayed on the client.

This procedure is not recommended. All of the X traffic between the client and server is sent unencrypted.