LAB – ESTABLISH A SSH CONNECTION

OBJECTIVE

In this lab, you will create a **pexpect** session object connecting via an SSH session to a network device with error checking for each step, modifying the hostname and then disconnect, showing the success or failure of the ability to connect via SSH.

PART 1

Copy the required configuration for this lab from flash into the running-configuration of the CSR1000v router.

STEP 1: COPY CONFIGURATION

In the CSR1kv router console, enter **privileged exec** mode and copy the additional configuration located in the **CONFIG4** file to the running-configuration using the commands:

```
CSR1kv> enable
CSR1kv# copy flash:CONFIG4 running-config
```

```
CSR1kv>enable
CSR1kv#copy flash:CONFIG4 running-config
Destination filename [running-config]?
95 bytes copied in 0.017 secs (5588 bytes/sec)
CSR1kv#
```

STEP 2: VERIFY CONFIGURATION

Verify the new running-configuration by using the command:

```
CSR1kv# show run | include secret
```

NOTE:
To get the pipe (**|**) command, you may need to use the keys **Shift** + **#** rather than your local keyboard input.

```
CSR1kv#show run | include secret
enable secret 5 $1$AJmo$U6W.jxkEI8TvyXjcXQ/p01
username cisco privilege 15 secret 5 $1$S2XU$IsZBWUig5GV9LM8YNvUaj.
username prne secret 5 $1$raVz$cwEfpBMP8Usr8EHwN5ab6.
CSR1kv#
```

PART 2

Open a terminal and switch to the lab directory

STEP 1: OPEN A TERMINAL WINDOW

Double-click the Terminal icon on the desktop to open the terminal window for use in this lab.

STEP 2: CHANGE DIRECTORY

Change to the directory **labs/prne/** in the user home directory, which holds the files for the course labs.

```
~$ cd labs/prne/
```

PART 3

Open **Visual Studio Code**, create a new file and save it with a filename of **establish-a-ssh-connection.py**, ensuring to save the file in the **~/labs/prne/** directory, as otherwise the code will require modification to find the associated files that are used.

STEP 1: IMPORT PEXPECT

Import the pexpect library.

```
# Import required modules/packages/library
import pexpect
```

STEP 2: DEFINE VARIABLES

Define variables for your IP address, username, and password and assign the following values:

- IP Address: 192.168.56.101
- Username: prne
- Password: cisco123!
- Enable password: class123!

```
# Define variables
ip_address = '192.168.56.101'
username = 'prne'
password = 'cisco123!'
password_enable = 'class123!'
```

STEP 3: CREATE SESSION

Create a session using the pexpect 'spawn' method, passing in a ssh command.

```
# Create the SSH session
session = pexpect.spawn('ssh ' + username + '@' + ip_address,
                        encoding='utf-8', timeout=20)
result = session.expect(['Password:', pexpect.TIMEOUT, pexpect.EOF])
```

STEP 4: CHECK FOR ERRORS

Check for error in connecting session and display any errors in terminal.

```
# Check for error, if exists then display error and exit
if result != 0:
    print('--- FAILURE! creating session for: ', ip_address)
    exit()
```

STEP 5: SEND CREDENTIALS

Send the user credentials to the device and check for any errors and display any in terminal.

```
# Session expecting password, enter details
session.sendline(password)
result = session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])

# Check for error, if exists then display error and exit
if result != 0:
    print('--- FAILURE! entering password: ', password)
    exit()
```

NOTE:
If you have not connected to the device through ssh previously and therefore do not have the keys, this will fail. As such it is recommended that you attempt to ssh from a terminal or PuTTY first to accept any keys and verify that it is working without issues.

STEP 6: CHALLENGE (OPTIONAL)

Rather than having to login manually and accept the ssh key through another terminal session, it is possible to auto accept the ssh keys, see if you can change the code to complete this.

If you have already accepted the key, then it can be removed using the command below from a terminal window:

```
~/labs/prne$ ssh-keygen -R 192.168.56.101
```

PART 4

Change the hostname of the router.

STEP 1: ENTER PRIVILEGED EXEC MODE

Change to Privileged EXEC mode on the router.

```
# Enter enable mode
session.sendline('enable')
result = session.expect(['Password:', pexpect.TIMEOUT, pexpect.EOF])

# Check for error, if exists then display error and exit
if result != 0:
    print('--- Failure! entering enable mode')
    exit()

# Send enable password details
session.sendline(password_enable)
result = session.expect(['#', pexpect.TIMEOUT, pexpect.EOF])

# Check for error, if exists then display error and exit
if result != 0:
    print('--- Failure! entering enable mode after sending password')
    exit()
```

STEP 2: ENTER CONFIGURATION MODE

Change to configuration mode on the router.

```
# Enter configuration mode
session.sendline('configure terminal')
result = session.expect([r'.\(config\)#', pexpect.TIMEOUT, pexpect.EOF])

# Check for error, if exists then display error and exit
if result != 0:
    print('--- Failure! entering config mode')
    exit()
```

STEP 3: CHANGE THE HOSTNAME

Change the hostname of the router to R1, verifying that it is correct.

```
# Change the hostname to R1
session.sendline('hostname R1')
result = session.expect([r'R1\(config\)#', pexpect.TIMEOUT, pexpect.EOF])

# Check for error, if exists then display error and exit
if result != 0:
    print('--- Failure! setting hostname')
```

STEP 4: EXIT CONFIGURATION AND PRIVILEDGED EXEC MODES

Before exiting the application, you should exit out of configuration mode and privileged EXEC mode.

```
# Exit config mode
session.sendline('exit')

# Exit enable mode
session.sendline('exit')
```

STEP 5: DISPLAY SUCCESS MESSAGE

Display a success message in the terminal if everything has worked.

```
# Display a success message if works
print('-------------------------------------------------------')
print('')
print('--- Success! connecting to: ', ip_address)
print('---               Username: ', username)
print('---               Password: ', password)
print('')
print('-------------------------------------------------------')
```

STEP 6: CLOSE THE SSH SESSION

Terminate the SSH session.

```
# Terminate SSH session
session.close()
```

STEP 7: SAVE, RUN AND VERIFY APPLICATION

Save you application and then run it from the terminal rather than from within visual studio code.

```
~/labs/prne$ python3 establish-a-ssh-connection.py
```

Verify that the ssh connection is successful and the output is comparable to below.

```
devasc@labvm:~/labs/prne$ python3 establish-a-ssh-connection.py
-------------------------------------------------------

--- Success! connecting to:  192.168.56.101
---              Username:  prne
---              Password:  cisco123!

-------------------------------------------------------
devasc@labvm:~/labs/prne$
```

NOTE:
If you are unable to connect to the router via your script and receive a response similar to below, it may be due to not having accepted the ssh keys. As such, ssh to the device from a terminal or PuTTY to accept any keys. Then re-run and verify that it is working without issues.

```
devasc@labvm:~/labs/prne$ python3 establish-a-ssh-Connection.py
--- FAILURE! creating session for:  192.168.56.101
```

PART 5 (OPTIONAL BUT HIGHLY RECOMMENDED)

As this lab is completed in NETLAB+ and your code files will be erased when the reservation ends, it is advisable to save your files in GitHub under your repository for this course.