Hands-On – Paramiko

Save your solutions in a docx file with name in this format: <mark>*cnit381_1_hw3_YOURNAME.docx*</mark> *and submit this file to Stout's Canvas.*

Provide codes (in text) for each Challenge with result-evidences (can be screenshots).

All output must be formatted UTF-8.

If your solution is not correct, then try to understand the error messages, check the slides again, post questions on Canvas, rewrite the solution and test it again. Repeat this step until you get the correct solution.

Save the solutions in files for future reference or recap.

I'd recommend you use my Virtual Lab and this topology in GNS3 (below) or create a similar one.
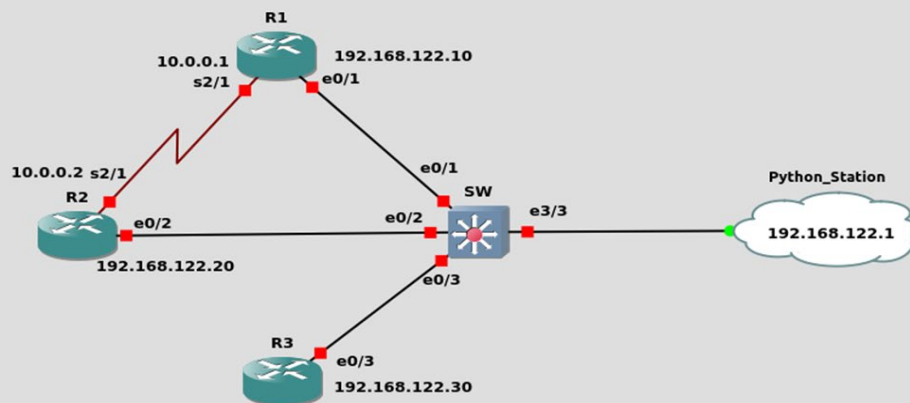
Challenge #1

Create a Python script that connects to a Cisco Router using SSH and Paramiko. The script should execute the *show run* command.

Print out the output of the command.

Challenge #2

Change the solution from Challenge #1 so that it will prompt the user for the SSH password securely (use getpass module).

Challenge #3

Change the solution from Challenge #1 so that it will save the output to a file instead of printing it.

Challenge #4

Consider the solution from Challenge #1

Change the connecting part so that the connect() method of the ssh_client can receive a **kwargs argument like this: ssh_client.connect(**router)

Challenge #5

Create a Python script that connects to a R1 using SSH and Paramiko, enters the enable mode and then executes show running-config.

The entire output should be saved to a file in the current working directory.

Challenge #6

A Network Engineer has created Paramiko Python script that executes "show ip interface brief" on a remote Cisco Router and displays the output.

Although the script connects and authenticates successfully, it doesn't display the entire output of the command, but only a part of it.

Your task is to troubleshoot the issue and solve it so that it displays the entire output.

Challenge #7

A Network Engineer has created a Paramiko Python script (part of the script is showing below) that executes "show ip interface brief" on a remote Cisco Router and displays the output.

```
shell.send('show ip int brief')
time.sleep(1)

output = shell.recv(100000)
# decoding from bytes to string
output = output.decode()
print(output)
```

However since there is a single error in the script it can't display the output of the command.

Your task is to troubleshoot the issue and solve it so that it works as expected.

Challenge #8

Create a Python file name *myParamiko.py*. The file contains a collection of helper functions:

1. connect(), will take ip address, port, username and password as arguments and return a paramiko.SSHClient object.
2. get_shell(), will take a paramiko.SSHClient object as arguments and return a shell object.
3. send_command(), will a shell object and a command (string) as arguments then send the command to the remote host.
4. show(), (), will a shell object, a command (string) as arguments then send the command to remote host and print out the return output.
5. close(), will take a paramiko.SSHClient object as arguments and close this connection.
6. get_list_from_file(), will take a filename (string) and return a list of remote-hosts that can be used by paramiko.SSHClient . The file's format will looks like below:
    [{'server_ip': '192.168.122.10', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'},
    {'server_ip': '192.168.122.20', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'},
    {'server_ip': '192.168.122.30', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'}]

Challenge #9

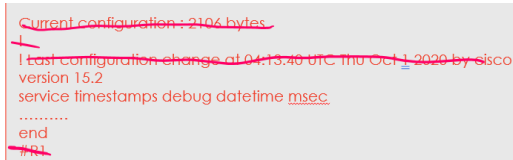Use the functions in *myParamiko.py* to configure OSPF on R1, R2, R3. The OSPF commands are:

"*router ospf 1, network 0.0.0.0 0.0.0.0 area 1, end, show ip protocols*".

Challenge #10

Use the functions in *myParamiko.py* to backup R1, R2, R3 configurations to files.

Requirements:

1. Remove unnecessary texts from the *show run* commands:
    a.
2. Save each router configuration in a separate file and apply version-control for the filenames

Hands-On – Netmiko

Challenge #1

Create a Python script that connects to R1 using SSH and Netmiko. The script should execute the commands: "*show ip int brief, show run, username u1 secret cisco*" sequentially by using the send_command() method of Netmiko.

Print out the output of the command.

Challenge #2

A Network Engineer has created Netmiko Python script that executes "*int lo 0*" command on a remote Cisco Router and displays the output.

Although the script connects, authenticates successfully and creates the loopback 0, it doesn't proceed any further, the script never stops.

Your task is to troubleshoot the issue and solve it so the script can run normally.

Challenge #3

Create 03 text-files that contain OSPF configurations:

192.168.122.10_ospf.txt

router ospf 1

router-id 1.1.1.1

net 0.0.0.0 0.0.0.0 area 0

distance 80

default-information originate

192.168.122.20_ospf.txt

router ospf 1

router-id 2.2.2.2

net 0.0.0.0 0.0.0.0 area 0

distance 80

default-information originate

192.168.122.30_ospf.txt

router ospf 1

router-id 3.3.3.3

net 0.0.0.0 0.0.0.0 area 0

distance 80

default-information originate

<mark>Create a router list file, namely *routers.txt* that contains:</mark>

```
[{
    'device_type': 'cisco_ios',
    'host': '192.168.122.10',
    'username': 'cisco',
    'password': 'cisco',
    'port': 22,
    'secret': 'cisco',
    'verbose': True
},
{
    'device_type': 'cisco_ios',
    'host': '192.168.122.20',
    'username': 'cisco',
    'password': 'cisco',
    'port': 22,
    'secret': 'cisco',
    'verbose': True
},
{
    'device_type': 'cisco_ios',
    'host': '192.168.122.30',
    'username': 'cisco',
    'password': 'cisco',
    'port': 22,
    'secret': 'cisco',
    'verbose': True
}]
```

<mark>Create a python script that: For each router in *routers.txt* file, use Netmiko to execute the list of commands in *192.168.122.XX_ospf.txt*, i.e. R1 will need file *192.168.122.10_ospf.txt*, R2 will need file *192.168.122.20_ospf.txt* and R3 will need file *192.168.122.30_ospf.txt.*</mark>

Challenge #4

Change the solution from Challenge #3 so that it will run in multi-threading.

Challenge #5

Use simplecrypt library (you may need to install it) to create a Python Script, *encrypt_dev_info.py* that contains 2 functions:

1. encrypt_file(), that take nothing and will prompt the user to enter an original filename, a key and an output filename. Then create an encrypted file from the original file and name it with the output filename.
2. decrypt_file(), will take a filename (string) as an argument then decrypt the file to a list and return that list.

Use the functions in *encrypt_dev_info.py* to encrypt *routers.txt* and name the output file as *encrypted_routers.txt*. You must use terminal to do this encryption task.

Challenge #6

Change the solution from **Challenge #4** so that it will use *encrypted_routers.txt* instead of *routers.txt.*

END.